

ADVANCED PREDICTION OF STUDENT PERFORMANCE IN A COLLEGE USING MACHINE LEARNING

A Project Report submitted to

**JAWAHARLAL NEHRU TECHNOLOGICAL UNIVERSITY
ANANTAPUR, ANANTHAPURAM**

In partial fulfillment of the requirements for the award of the degree of

**BACHELOR OF TECHNOLOGY
In
COMPUTER SCIENCE AND ENGINEERING**

Submitted by

Vaddi Vijayalakshmi	198R1A0538
Omitta Manjunath	198R5A0522
Posa Supraja	208R5A0502
Chinnakathera GuruMohan	198R1A0506

Under the esteemed guidance of

Mr. Shaik Khaleel M.Tech

Assistant professor



**DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING
SAI RAJESWARI INSTITUTE OF TECHNOLOGY**

AN ISO 9001:2015 CERTIFIED INSTITUTION

(Approved by AICTE, New Delhi and Affiliated to J.N.T.U.A. Ananthapuramu)

Lingapuram (V), Proddatur, Kadapa(Dist.)-516360

2019-2023



SAI RAJESWARI INSTITUTE OF TECHNOLOGY

AN ISO 9001:2015 CERTIFIED INSTITUTION

(Approved by AICTE, New Delhi and Affiliated to J.N.T.U.A. Ananthapuramu)

Lingapuram (V), Proddatur, Kadapa (Dist.)-516360,A.P

DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING

Certificate

This is to certify that the Project Report entitled

ADVANCED PREDICTION OF STUDENT PERFORMANCE IN A COLLEGE USING MACHINE LEARNING

is the bonafide work done and

Submitted by

Vaddi Vijayalakshmi	198R1A0538
Omitta Manjunath	198R5A0522
Posa Supraja	208R5A0502
Chinnakathera GuruMohan	198R1A0506

In the Department of Computer Science and Engineering, sai Rajeswari Institute of Technology, Proddatur affiliated to J.N.T.U.A., Ananthapuramu in partial fulfillment of the requirements for the award of Bachelor of Technology in Computer Science and Engineering during Academic Year 2022-2023.

Submitted on: _____

Internal Guide

HOD

Mr. S. Khaleel, M.Tech
Assistant professor

Dr. G. RAMASUBBA REDDY, M.E, Ph. D
Professor

Internal Examiner

External Examiner

ACKNOWLEDGEMENT

Any achievement, be it scholastic or otherwise does not depend solely on the individual effort but the guidance, encouragement, and cooperation of intellectuals, elders, and friends. We would like to take this opportunity to thank them all.

We wholeheartedly express our gratitude and esteemed regards to Project guide **Mr. SHAIK.KHALEEL, M.Tech.**, in the Department of Computer Science and Engineering, for providing us his valuable guidance and inspiration in carrying out our project studies. His constant support and encouragement enabled us to complete this work successfully.

We express our sincere thanks to Project Co-Ordinator **Mr.M.Stanlywit, M.Tech**, Assistant Professor in the Department of Computer Science and Engineering, for providing us his valuable guidance and inspiration in carrying out our project studies.

With a deep sense of gratitude, we acknowledge **Dr. G. RAMASUBBA REDDY**, M.E, Ph.D., Head of the Dept., Computer Science & Engineering, for his valuable support and help in completing our project successfully.

We express our sincere thanks to **Dr. PANDURANGAN RAVI**, M.Tech, Ph.D., our beloved Principal, for his encouragement and suggestions during our course of study.

We feel honored for placing our warm salutation to **THE MANAGEMENT** Sai Rajeswari Institute of Technology, Proddatur, which allowed us to obtain a strong base in B. Tech and profound knowledge.

Finally, we would like to express our sincere thanks to all the **Faculty Members** of the C.S.E Department, Lab Technicians, Friends & Family members, and all, who have helped us to complete this work successfully.

V.Vijaya lakshmi (198R1A0538)
O.Manjunath Reddy(198R5A0522)
P.Supraja(208R5A0502)
C.GuruMohan(198R1A0506)

CONTENTS

Chapter No	Chapter Name		Page No
	List of Diagrams		
	List of Abbreviation		
	Abstract		
1	Introduction		1
2	System Analysis		2
	2.1	Existing System	2
	2.2	Disadvantages of Existing System	2
	2.3	Proposed System	3
	2.4	Advantages of Proposed System	3
3	Methodology and Algorithms		5-8
4	System Requirement Specifications		9-10
	4.1	Software Requirements	9
	4.2	Hardware Requirements	10
5	Feasibility Study		11
	5.1	Feasibility Study	11
6	System Design		12
	6.1	Design Approaches	12
	6.1.1	Class Diagram	13-14
	6.1.2	Use case Diagram	14-15
	6.1.3	Sequence Diagram	15-16
	6.1.4	Deployment Diagram	16-17
7	System Coding		18
	7.1	Sample Code	18-24
	7.2	About Programming Language and arduino	25-39
8	Implementation		40
	8.1	Modules	40-41
9	Testing		42-44
10	Results		45-49
11	Conclusion		50
	Future Enhancement		50
	Bibliography		51-52
	References		53

LIST OF DIAGRAMS

S.NO	Figure no	Name	Page no
1	6.1.1	Class diagram	16
2	6.1.2	Use case Diagram	17
3	6.1.3	Sequence diagram	18
4	6.1.8	Deployment Diagram	21

LIST OF ABBREVIATIONS

S.NO	Abbreviation	Description
1	KNN	K nearest neighbour
2	SVM	Support vector machine
3	OS	Operating System
4	UML	Unified modelling language
5	GSM	Global system for mobile communication
6	GUI	Graphical user interface

ABSTRACT

Education is an important element of the society, every government and country in the world work so hard to improve this sector. The educational systems have been affected in many ways; studies show that student's performance has decreased since then, which highlights the need to deal with this problem more seriously and try to find effective solutions, as well as the influencing factors. The educational systems need, at this specific time, innovative ways to improve quality of education to achieve the best results and decrease the failure rate. In order for an institute to provide quality education to learners, deep analysis of previous records of the learners can play a vital role, and wanted to work on this challenging task. As already mentioned, with the help of the old students records, we can come up with a model that can help students improve their performance in exams by predicting the student success. So, it is obvious it's a problem of classification, and we will classify a student based on his given informations, and we will also use different classifiers such as KNN or SVM classifier and compare between them.

Keywords: KNN, SVM, GSM

CHAPTER-1

INTRODUCTION

Higher education is increasingly seen as essential by many employers. This has led to an increase, not only in the number of people opting to pursue degrees in all fields, but also in the diversity of the students. While in the past students were likely to be affluent, young males seeking an academic career in a specific field, the backgrounds and motivations of students today are significantly more diverse. Due to the increase in student numbers and diversity it can be difficult for an advisor of studies to get to know each student personally and provide meaningful feedback on their academic performance. This can ultimately lead to student dissatisfaction and an increase in the number of students failing to complete their degree on time or dropping out of university.

By identifying students at risk of failing early and providing feedback at an early stage in a degree it is believed that student performance can be improved and student retention rates can be increased, outcomes which benefit both the students and the institutions. A tool or application, made available to students and lecturers, which can take in student information and provide information of their expected performance and provide feedback on how to improve their current situation would be greatly beneficial in this regard.

Technological advancements in recent years have made computationally expensive prediction methods such as machine learning a viable option for a variety of applications such as this and, with a great deal of research having gone into the development and improvement of machine learning algorithms, there are several algorithms available which are suitable for this specific task. Identifying a suitable algorithm is the first step in developing an application which can be used as means of properly engaging with students who would benefit most from engagement from lecturers.

CHAPTER – 2

SYSTEM ANALYSIS & FEASIBILITY STUDY

2.1. Existing Method:

Comparative analysis among different data mining algorithm for attribute selection and classification was conducted on this two phase study which aimed to predict the students' performance in Java Programming and be able to generate recommendations. Processes in Knowledge Discovery in Database (KDD) was followed in finding patterns among the historical data. Logistic Regression and Correlation based Feature Selection was used for finding significant predictors. Classifiers such as CHAID, Exhaustive CHAID, CRT, QUEST, J48, BayesNet, NaïveBayes and JRip were implemented and it was found out that J48, on the context of this study, has the most straightforward rules set and the highest percentage of prediction. For the second phase evolutionary prototyping implemented through Ruby on Rails was done to develop a webbased examination module that will determine the students' index of learning style and to assess their prior knowledge in Java. A course content recommendation presenting the learners' strengths and weaknesses in the subject with suggested method of learning style will be automatically generated by the system.

2.2. Disadvantages:

- It is very timetaking.
- The complexity increases tending to a very high probability of error.
- Difficulties in handling errors.

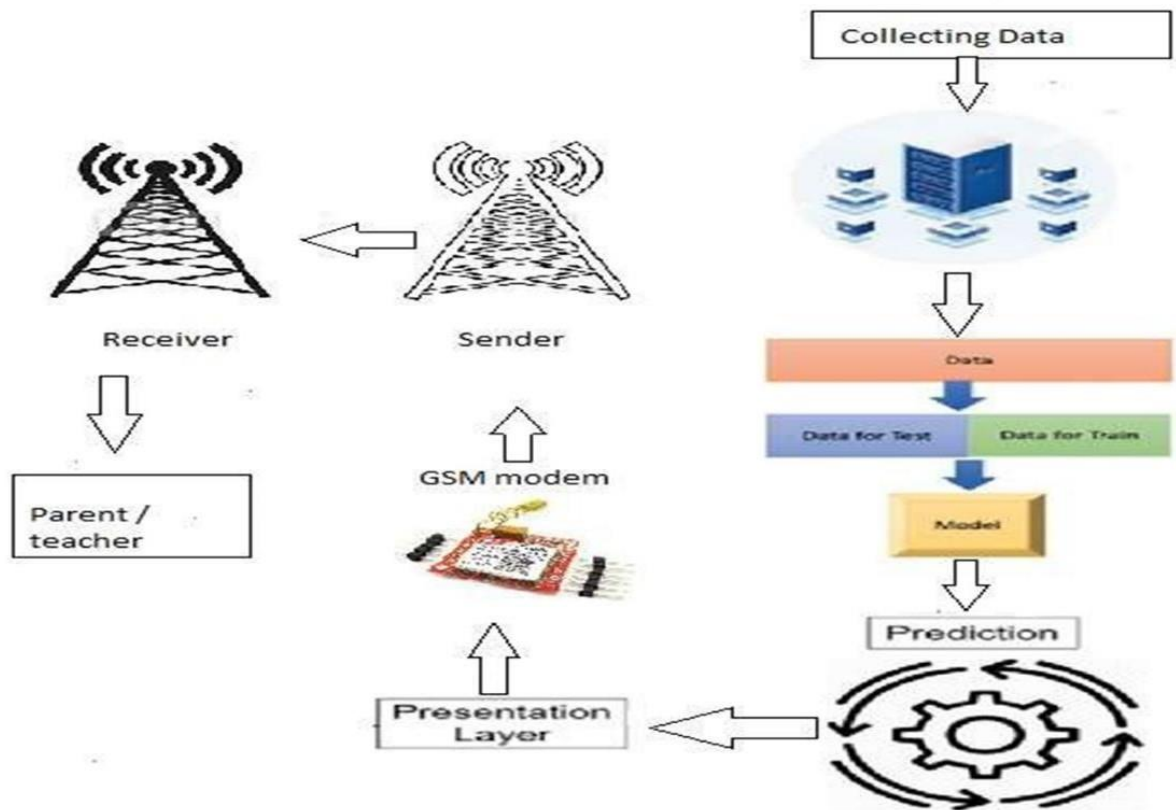
2.3. Proposed System:

The Proposed System of Prediction of student performance is based on the Machine Learning with the help of the old students records, we can come up with a model that can let us help students improve their performance in exams by predicting the student success. So, it is obvious it's a problem of classification, and we will classify a student based on his given informations, and we will also use different classifiers such as KNN or SVM classifier and compare between them. Many factors affect a student performance in exams like family problems or alcohol consumption, and by using our skills in machine learning we want to : predict whether a student will pass his final exam or not come up with the best classifier that is more accurate and avoid overfitting and underfitting by using simple techniques.

2.4. Advantages:

- Able to finding all difficulties of student failures.
- Introducing more efficient algorithms.
- Easy to code.
- Thoroughly predicting the student performance.
- Sent notifications to the mentors.

Architecture



CHAPTER – 3

METHODOLOGY AND ALGORITHMS

K-Nearest Neighbours

K-Nearest Neighbours is one of the simplest and most popular tuneable algorithms used to solve classification problems (Cunningham & Delany, 2007). Classification of an example is carried out by plotting the example to be classified against all the training data and determining the nearest training data set examples, a weight can then be assigned to each of these neighbours which determines how influential each training example is – this weight is usually determined by the distance to the neighbour – or each neighbour can be considered to have an equal weight regardless of distance, a majority vote on each of the nearest neighbours will then determine the predicted class. K-Nearest Neighbours is considered an instance-based algorithm as no training is carried out prior to making a prediction, instead the training data is stored in a database and queried when a prediction is required. This reduces the training time but can result in longer prediction times, especially when using data sets with large numbers of features or with many training examples.

One of K-Nearest Neighbours' most important parameters for use in tuning is k , which is the number of neighbours which will be identified and used when attempting to classify an example. By increasing k the variance will be reduced while the bias is increased which will reduce the complexity of the model and reduce overfitting, however increasing k by too much will result in underfitting due to the increase in bias so it is important to find a value for k which does not increase bias too much. The best value of k will differ from data set to data set and will require investigation to determine.

Figure 1 shows an example of a prediction being carried out. In this example k is equal to five which will result in the five nearest neighbours being considered. It can be seen that of these five, four are in Class 1 while only one is in Class 2. Assuming in this case that the weighting is uniform, the majority vote will be Class 1.

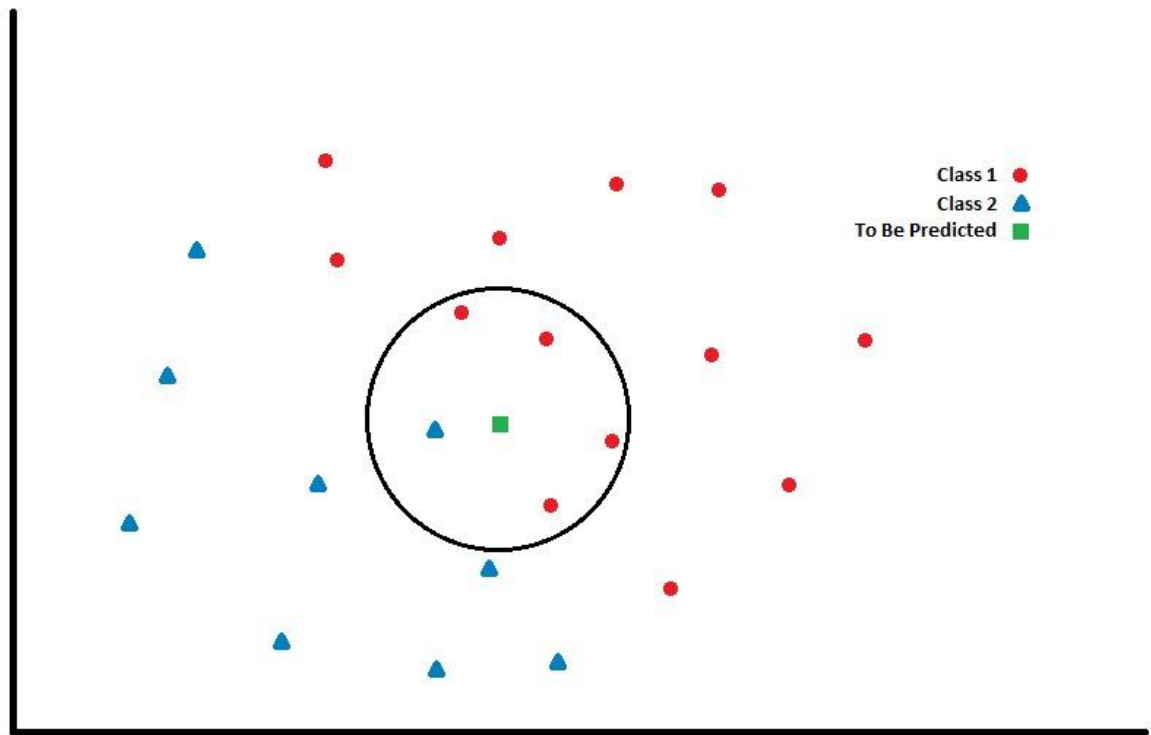


Figure 1. K-Nearest Neighbour Example

K-Nearest Neighbours tends to work best with features which are continuous as the geographical location of each example will tend to vary more. This is ideal for grade data where a percentage mark is given but is likely to be more problematic for examples containing discrete features such as gender. For this reason, it will be important to remain aware of the effect student background features have on the performance of the model.

Support Vector Machine

SVM (Support Vector Machine) is a machine learning algorithm which is more complex than the likes of Naïve Bayes and K-Nearest Neighbours but is no less popular. The early stages of classification with SVM are similar to K-Nearest Neighbour in that the training examples are initially plotted in an n -dimensional plot where n is the number of available features in the data set. The SVM algorithm then determines a linear $(n-1)$ -dimensional hyperplane, or decision boundary, which splits the training data in such a way that examples from each class are on separate sides of the decision boundary while maximising the distance between the boundary and the nearest points from each class. Figure 2 shows an example of a binary classification problem

where the maximisation of the margin between the nearest examples has been sacrificed in order to prevent any classification errors.

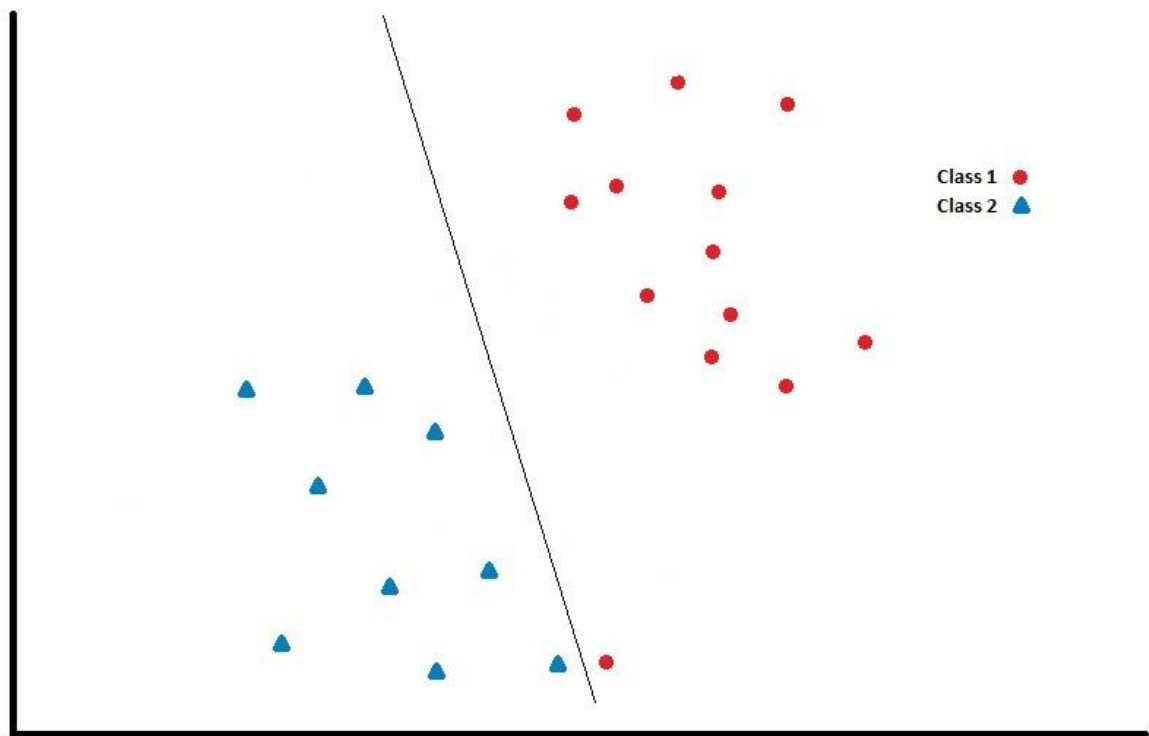


Figure 2. Support Vector Machine Hyperplane

In a real situation it, more often than not isn't possible to separate the examples into classes perfectly and so the algorithm seeks first to determine the best fit decision boundary which maximises accuracy. To overcome this an error tolerance is introduced which allows for the model to tolerate misclassified examples to produce a more suitable margin.

SVM can also produce a nonlinear decision boundary through the use of a method known as Kernel Trick. This can transform existing features into new features which then allows for a curved decision boundary and margin. The Kernel Trick which will be employed in this study is RBF (Radial Basis Function) which curves the decision boundary by allowing features to have an influence over it, the influence each example has is determined by the position of the decision boundary and the position of that example, essentially weighting each example when making a classification. The weight of each individual training example can also be changed using the gamma parameter.

From the previous studies it can be seen that SVM performs well in a variety of situations, often outperforming other classifiers it is compared against.

SVM is an effective classifier when there is a distinct margin between training data classes which may be the case with student grade data. SVM also performs better with smaller data sets, however large datasets most heavily affect the time taken to train and, while time taken to train should be considered when selecting a model, in this experiment it is not the important, which work well with the data sets available. There are drawbacks to SVM in that performance is generally better with large numbers of features, where the available data sets have less than twenty features for this experiment.

CHAPTER – 4

SYSTEM REQUIREMENTS SPECIFICATION

Functional and non-functional requirements:

Requirement's analysis is very critical process that enables the success of a system or software project to be assessed. Requirements are generally split into two types: Functional and non-functional requirements.

Functional Requirements: These are the requirements that the end user specifically demands as basic facilities that the system should offer. All these functionalities need to be necessarily incorporated into the system as a part of the contract. These are represented or stated in the form of input to be given to the system, the operation performed and the output expected. They are basically the requirements stated by the user which one can see directly in the final product, unlike the non-functional requirements.

Non-functional requirements: These are basically the quality constraints that the system must satisfy according to the project contract. The priority or extent to which these factors are implemented varies from one project to other. They are also called non-behavioral requirements.

They basically deal with issues like:

- Portability
- Security
- Maintainability
- Reliability
- Scalability
- Performance
- Reusability
- Flexibility

4.1.HARDWARE AND SOFTWARE REQUIREMENTS:

Hardware:

Operating system : Windows 7 or 7+
RAM : 8 GB



Hard disc or SSD : More than 500 GB
Processor : Intel 3rd generation or high or Ryzen with 8 GB Ram

Software:

Software's : Python 3.6 or high version
IDE : Jupyter Notebook

CHAPTER – 5

FEASIBILITY STUDY

The feasibility of the project is analysed in this phase and business proposal is put forth with a very general plan for the project and some cost estimates. During system analysis the feasibility study of the proposed system is to be carried out. This is to ensure that the proposed system is not a burden to the company. For feasibility analysis, some understanding of the major requirements for the system is essential.

Three key considerations involved in the feasibility analysis are

- ◆ ECONOMICAL FEASIBILITY
- ◆ TECHNICAL FEASIBILITY
- ◆ SOCIAL FEASIBILITY

Economic feasibility:

This study is carried out to check the economic impact that the system will have on the organization. The amount of fund that the company can pour into the research and development of the system is limited. The expenditures must be justified. Thus, the developed system as well within the budget and this was achieved because most of the technologies used are freely available. Only the customized products had to be purchased.

Technical feasibility:

This study is carried out to check the technical feasibility, that is, the technical requirements of the system. Any system developed must not have a high demand on the available technical resources. This will lead to high demands on the available technical resources. This will lead to high demands being placed on the client. The developed system must have a modest requirement, as only minimal or null changes are required for implementing this system.

Social feasibility:

The aspect of study is to check the level of acceptance of the system by the user. This includes the process of training the user to use the system efficiently. The user must not feel threatened by the system, instead must accept it as a necessity. The level of acceptance by the users solely depends on the methods that are employed to educate the user about the system and to make him familiar with it. His level of confidence must be raised so that he is also able to make some constructive criticism, which is welcomed, as he is the final user of the system.

CHAPTER - 6

SYSTEM DESIGN

6.1. Design Approaches

Input Design:

In an information system, input is the raw data that is processed to produce output. During the input design, the developers must consider the input devices such as PC, MICR, OMR, etc.

Therefore, the quality of system input determines the quality of system output. Well-designed input forms and screens have following properties –

- It should serve specific purpose effectively such as storing, recording, and retrieving the information.
- It ensures proper completion with accuracy.
- It should be easy to fill and straightforward.
- It should focus on user's attention, consistency, and simplicity.
- All these objectives are obtained using the knowledge of basic design principles regarding –
 - What are the inputs needed for the system?
 - How end users respond to different elements of forms and screens.

Objectives for Input Design:

The objectives of input design are –

- To design data entry and input procedures
- To reduce input volume
- To design source documents for data capture or devise other data capture methods
- To design input data records, data entry screens, user interface screens, etc.
- To use validation checks and develop effective input controls.

Output Design:

The design of output is the most important task of any system. During output design, developers identify the type of outputs needed, and consider the necessary output controls and prototype report layouts.

Objectives of Output Design:

The objectives of input design are:

- To develop output design that serves the intended purpose and eliminates the production of unwanted output.
- To develop the output design that meets the end user's requirements.
- To deliver the appropriate quantity of output.
- To form the output in appropriate format and direct it to the right person.
- To make the output available on time for making good decisions.

6.1.1. CLASS DIAGRAM

The class diagram depicts a static view of an application. It represents the types of objects residing in the system and the relationships between them. A class consists of its objects, and also it may inherit from other classes. A class diagram is used to visualize, describe, document various different aspects of the system, and also construct executable software code.

It shows the attributes, classes, functions, and relationships to give an overview of the software system. It constitutes class names, attributes, and functions in a separate compartment that helps in software development. Since it is a collection of classes, interfaces, associations, collaborations, and constraints, it is termed as a structural diagram.

The main purpose of class diagrams is to build a static view of an application. It is the only diagram that is widely used for construction, and it can be mapped with object-oriented languages. It is one of the most popular UML diagrams. Following are the purpose of class diagrams given below:

1. It analyses and designs a static view of an application.
2. It describes the major responsibilities of a system.
3. It is a base for component and deployment diagrams.
4. It incorporates forward and reverse engineering.

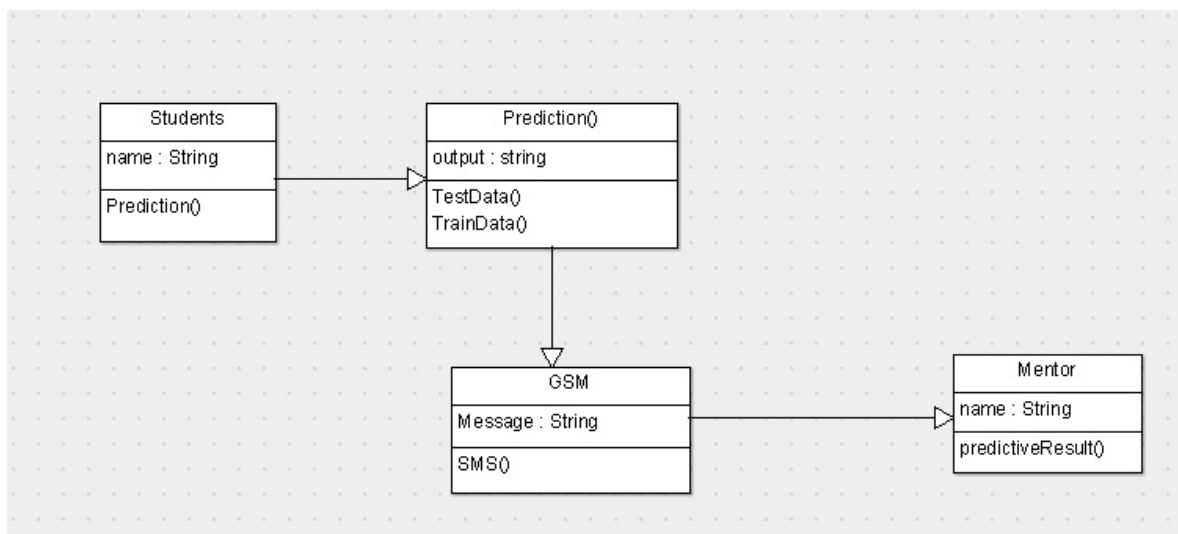


Fig:Class Diagram

6.1.2. USE CASE DIAGRAM

A use case diagram is used to represent the dynamic behavior of a system. It encapsulates the system's functionality by incorporating use cases, actors, and their relationships. It models the tasks, services, and functions required by a system/subsystem of an application. It depicts the high-level functionality of a system and also tells how the user handles a system.

Following are the purposes of a use case diagram given below:

1. It gathers the system's needs.
2. It depicts the external view of the system.
3. It recognizes the internal as well as external factors that influence the system.
4. It represents the interaction between the actors.

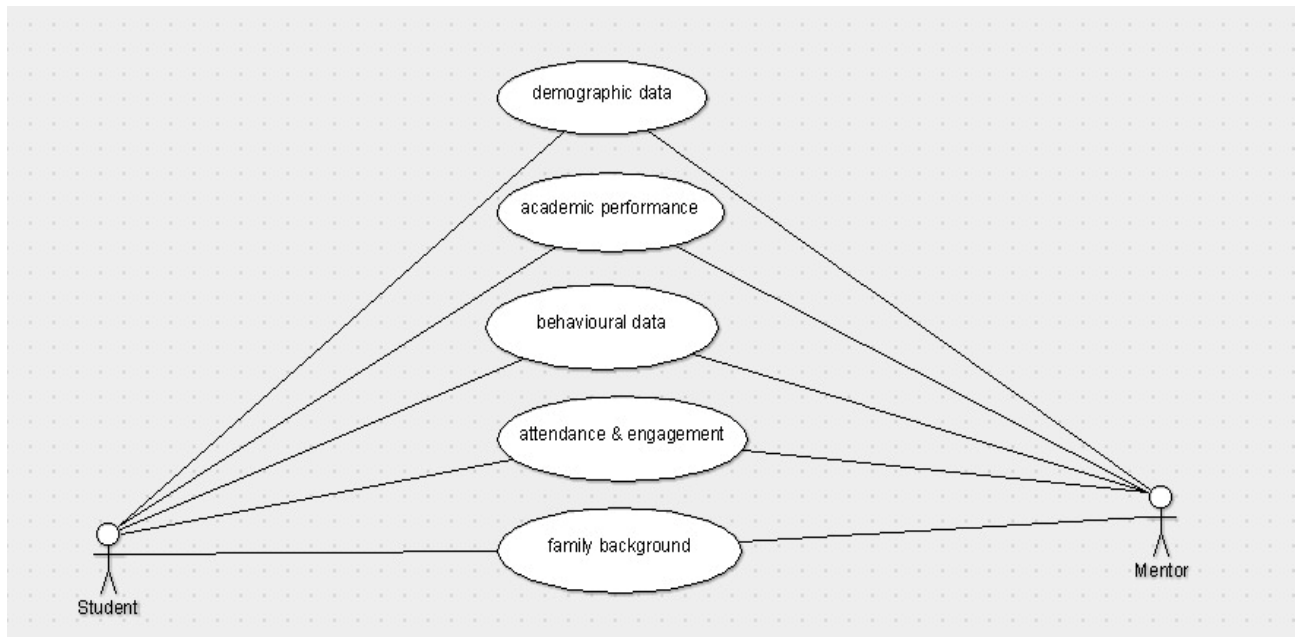


Fig: Usecase Diagram

6.1.3. SEQUENCE DIAGRAM

The sequence diagram represents the flow of messages in the system and is also termed as an event diagram. It helps in envisioning several dynamic scenarios. It portrays the communication between any two lifelines as a time-ordered sequence of events, such that these lifelines took part at the run time. In UML, the lifeline is represented by a vertical bar, whereas the message flow is represented by a vertical dotted line that extends across the bottom of the page. It incorporates the iterations as well as branching.

1. To model high-level interaction among active objects within a system.
2. To model interaction among objects inside a collaboration realizing a use case.

3. It either models generic interactions or some certain instances of interaction.

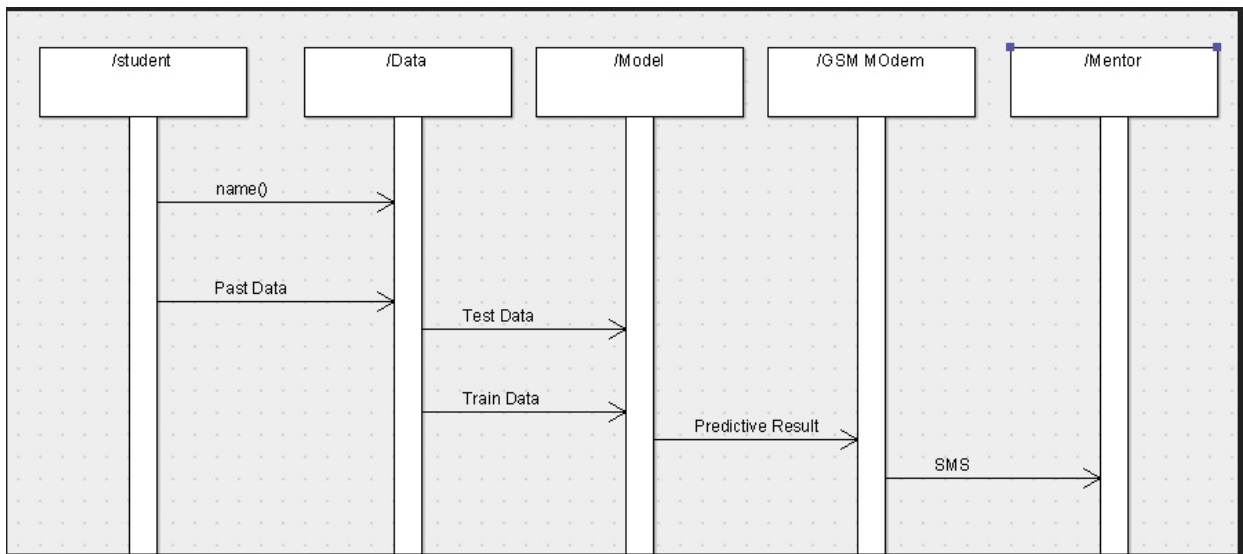


Fig:Sequence Diagram

6.1.4. DEPLOYMENT DIAGRAM

The deployment diagram visualizes the physical hardware on which the software will be deployed. It portrays the static deployment view of a system. It involves the nodes and their relationships.

It ascertains how software is deployed on the hardware. It maps the software architecture created in design to the physical system architecture, where the software will be executed as a node. Since it involves many nodes, the relationship is shown by utilizing communication paths.

The main purpose of the deployment diagram is to represent how software is installed on the hardware component. It depicts in what manner a software interacts with hardware to perform its execution.

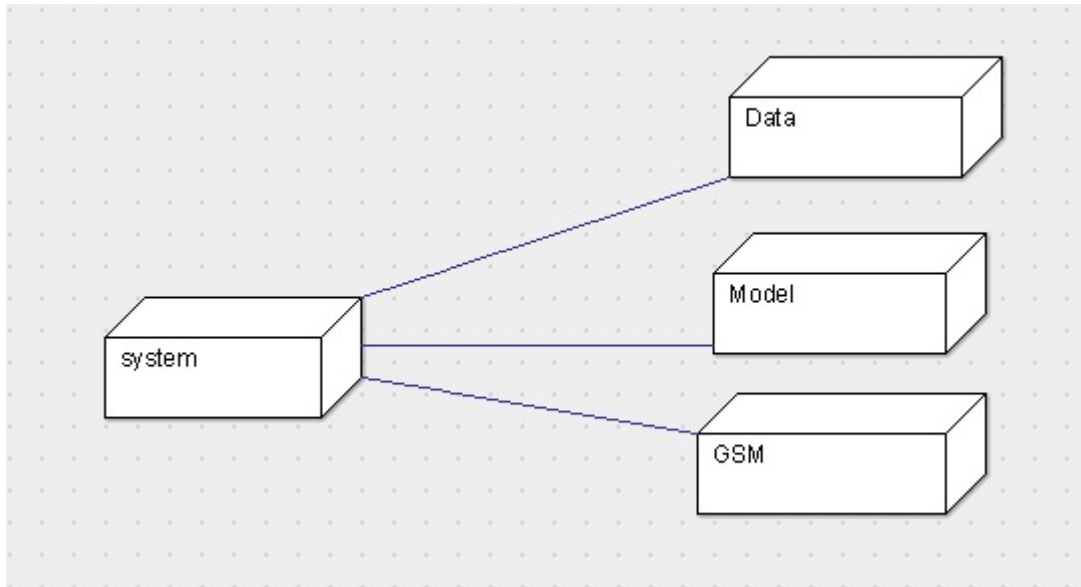


Fig:Deployment Diagram

CHAPTER – 7

SYSTEM CODING

6.1. Sample code

App.py

```
import numpy as np
import pandas as pd
def numerical_data(df):
    df['sex'] = df['sex'].map({'M': 0, 'F': 0})
    df['age']=0
    df['address'] = df['address'].map({'U': 0, 'R': 0})
    df['famsize'] = df['famsize'].map({'LE3': 0, 'GT3': 0})
    df['Pstatus'] = df['Pstatus'].map({'T': 0, 'A': 0})
    df['Mjob'] = df['Mjob'].map({'teacher': 0, 'health': 1, 'services': 2, 'at_home': 3, 'other': 4})
    df['Fjob'] = df['Fjob'].map({'teacher': 0, 'health': 1, 'services': 2, 'at_home': 3, 'other': 4})
    df['reason'] = df['reason'].map({'home': 0, 'reputation': 0, 'course': 0, 'other': 0})
    df['guardian'] = df['guardian'].map({'mother': 0, 'father': 0, 'other': 0})
    df['schoolsup'] = df['schoolsup'].map({'no': 0, 'yes': 0})
    df['famsup'] = df['famsup'].map({'no': 0, 'yes': 0})
    df['paid'] = df['paid'].map({'no': 0, 'yes': 0})
    df['activities'] = df['activities'].map({'no': 0, 'yes': 0})
    df['nursery'] = df['nursery'].map({'no': 0, 'yes': 0})
    df['higher'] = df['higher'].map({'no': 0, 'yes': 1})
    df['internet'] = df['internet'].map({'no': 0, 'yes': 1})
    df['romantic'] = df['romantic'].map({'no': 0, 'yes': 1})
    df['passed'] = df['passed'].map({'no': 0, 'yes': 0})
    df['traveltime']=0
    df['famrel']=0
    df['Dalc']=0
    df['Walc']=0

    # reorder dataframe columns :
    col = df['passed']
    del df['passed']
    df['passed'] = col
```

feature scaling will allow the algorithm to converge faster, large data will have same scal

```
def feature_scaling(df):
```

```
    for i in df:
```

```
        col = df[i]
```

```
        # let's choose columns that have large values
```

```
        if(np.max(col)>6):
```

```
            Max = max(col)
```

```
            Min = min(col)
```

```
            mean = np.mean(col)
```

```
            col = (col-mean)/(Max)
```

```
            df[i] = col
```

```
        elif(np.max(col)<6):
```

```
            col = (col-np.min(col))
```

```
            col /= np.max(col)
```

```
            df[i] = col
```

```
def read_file(file_path):
```

```
    l=[]
```

```
    df = pd.read_csv(file_path)
```

```
    dfv = pd.read_csv(file_path)
```

```
    numerical_data(df)
```

```
for col in df.columns:
```

```
    print(col)
```

```
d=dict(df.iloc[49])
```

```
print(d['Rollno'])
```

```
s='198R1A0550'
```

```
k1=0
```

```
k2=0
```

```
for i in d.items():
```

```
    if i[1]!=s:
```

```
        if i[0]=='gout' or i[0]=='absences' or i[0]=='romantic':
```

```
            k1=k1+i[1]
```

```
        else:
```

```
            k2=k2+int(i[1])
```

```
k=k2-k1
```

```
if k in range(20,30):
```

```
    print(s+" GOOD-PASSED")
```

```
elif k in range(10,20):
```

```

    print(s+" Average-Passed")
else:
    print(s+" Not good")
global newdict
newdict={ }
print("-----Student performance prediction-----")
gs=0
asum=0
ns=0
for i in range(50):
    p=dict(df.iloc[i])
    s=p['Rollno']
    k1=0
    k2=0
    for j in p.items():
        if j[1]!=str(s):
            if j[0]=='gout' or j[0]=='absences' or j[0]=='romantic':
                k1=k1+j[1]
            else:
                k2=k2+j[1]
    k=k2-k1
    if k in range(20,30):
        print(s+" GOOD-PASSED")
        newdict[s]='GOOD-PASSED'
        gs+=1
    elif k in range(10,20):
        print(s+" Average-Passed")
        newdict[s]='Average-Passed'
        asum+=1
    else:
        print(s+" Not good")
        newdict[s]='Not good'
        ns+=1
gsp=(gs/50)*100
avgp=(asum/50)*100
nsp=(ns/50)*100
l.append(("good students percentage",gsp))
l.append(("average students percentage",avgp))
l.append(("not good students percentage",nsp))
return l

```

```

import tkinter as tk
from tkinter import filedialog
from PIL import Image, ImageTk
def roll(input_value):
    for i in newdict.items():
        if i[0]==input_value:
            output_text.delete("1.0", tk.END) # Clear the output text box
            output_text.insert(tk.END, i[1]) # Display the output in the text box
            break
        else:
            output_text.delete("1.0", tk.END) # Clear the output text box
            output_text.insert(tk.END, "Invalid Choice")

def submit_input():
    input_value = input_entry.get()
    roll(input_value)

def choose_file():
    file_path = filedialog.askopenfilename()
    n = file_path.split('/')
    file_text.insert(tk.END, f"{n[-1]} is selected")
    #output_text.delete("1.0", tk.END) # Clear the output text box
    #output_text.insert(tk.END, i[1])
    if file_path:
        global msg
        read_file(file_path)
        file_button.pack_forget()
        input_label.pack()
        input_label.place(relx=0.3, rely=0.65, anchor=tk.W)
        input_entry.pack()
        input_entry.place(relx=0.45, rely=0.65, anchor=tk.W)
        submit_button.pack()
        submit_button.place(relx=0.4, rely=0.73, anchor=tk.W)
        output_text.pack()
        output_text.place(relx=0.32, rely=0.85, anchor=tk.W)
        sub_title.pack()
        sub_title.place(relx=0.53, rely=0.77, anchor=tk.W)
        output_text2.pack()
        output_text2.place(relx=0.53, rely=0.88, anchor=tk.W)
        msg=read_file(file_path)

```

```

    for i in read_file(file_path):
        for j in i:
            output_text2.insert(tk.END,str(j)+" ")
        output_text2.insert(tk.END,"\n")

# Create the main window
window = tk.Tk()
window.attributes("-fullscreen", True) # Set the window to full screen
window.title("My App") # Set the title of the window

# Set the background image
bg_image = tk.PhotoImage(file="srit.png")
bg_label = tk.Label(window, image=bg_image)
bg_label.place(relwidth=1, relheight=0.7)

# Add a minimize button
minimize_button = tk.Button(window, text="_", command=lambda: window.iconify())
minimize_button.place(relx=0.95, y=0, anchor="ne",relwidth=0.05)

# Add a close button
close_button = tk.Button(window, text="X", command=window.destroy)
close_button.place(relx=1.0, y=0, anchor="ne", relwidth=0.05)

header_frame = tk.Frame(window,width=200, height=100, borderwidth=2, relief="sunken")
header_frame.pack(pady=20) # Add some padding at the top

# Create a label for the header
header_label = tk.Label(header_frame, text="Student Performance Prediction", font=("Comic
Sans MS", 24),fg="white",bg="black")
header_label.pack()

text_widget = tk.Text(window, wrap=tk.WORD, height=8,
width=30,bg="#C54B8C",font=("Comic Sans MS", 15),fg="white")
text_widget.place(relx=0.75, rely=0.75, anchor=tk.W)

# Insert text into the text widget

```

```
text_widget.insert(tk.END, "INSTRUCTIONS\nstep-1.click on choose file button and select csv\nfile\nstep-2.enter roll number and click submit\nstep-3.Enter roll numbers as many as you need\nto predict th students performance")
```

```
# Create a "Choose File" button that opens a file dialog box
file_button = tk.Button(window, text="Choose File", font=("Comic Sans MS", 12),
command=choose_file, bg="white")
file_button.place(relx=0.05, rely=0.65, anchor=tk.W)
```

```
# Create a label to prompt the user for input
input_label = tk.Label(window, text="Enter Roll number:", font=("Comic Sans MS", 12),
fg="black")
```

```
# Create an entry field for the user to input a value
input_entry = tk.Entry(window, font=("Comic Sans MS", 12),fg="red")
```

```
# Create a "Submit" button that calls the submit_input() function when clicked
submit_button = tk.Button(window, text="Submit", font=("Comic Sans MS", 12),
command=submit_input, bg="blue",fg="white")
```

```
# Create a text box for the output
output_text = tk.Text(window, height=2, width=30, font=("Comic Sans MS", 12),
bg="white",fg="green")
sub_title=tk.Button(window,text="Overall predicted performance",font=("Comic Sans MS", 12),
bg="#8C7A77",fg="white")
output_text2 = tk.Text(window, height=5, width=28, font=("Comic Sans MS", 12),
bg="white",fg="#800080")
```

```
file_text = tk.Text(window, height=2, width=30,font=("Comic Sans MS", 12),
bg="white",fg="blue")
file_text.place(relx=0.05, rely=0.75, anchor=tk.W)
```

```
# Make all widgets corners round
input_label.config(relief=tk.RIDGE, bd=4, bg="white", borderwidth=4, highlightthickness=4,
highlightbackground="white", highlightcolor="white")
input_entry.config(relief=tk.RIDGE, bd=4, bg="white", borderwidth=4, highlightthickness=4,
highlightbackground="white", highlightcolor="white")
submit_button.config(relief=tk.RIDGE, bd=4, bg="blue", borderwidth=4, highlightthickness=4,
highlightbackground="blue", highlightcolor="white")
```

```

output_text.config(relief=tk.RIDGE, bd=4, bg="white", borderwidth=4, highlightthickness=4,
highlightbackground="white", highlightcolor="white")
sub_title.config(relief=tk.RIDGE, bd=4, bg="#8C7A77", borderwidth=4, highlightthickness=4,
highlightbackground="white", highlightcolor="white")
output_text2.config(relief=tk.RIDGE, bd=4, bg="#F5F5F5", borderwidth=4,
highlightthickness=4, highlightbackground="white", highlightcolor="white")
file_button.config(relief=tk.RIDGE, bd=4, bg="green", borderwidth=4, highlightthickness=4,
highlightbackground="white", highlightcolor="white")

```

Start the main loop to display the window and handle events

```

window.mainloop()
import serial
import time
arduino=serial.Serial(port='COM7',baudrate=9600,timeout=.1)
def write_read(x):
    print(x)
    arduino.write(bytes(x,'utf-8'))
    time.sleep(0.05)
    print("sent successfully")
while True:
    s="good"+str(msg[0][1])+"Average"+str(msg[1][1])+"Not good"+str(msg[2][1])
    write_read(s)
    time.sleep(10)
arduino.close()

```

6.3. About programming Language

Introduction to python

✓ Python

What Is a Script?

Up to this point, I have concentrated on the interactive programming capability of Python. This is a very useful capability that allows you to type in a program and to have it executed immediately in an interactive mode

Scripts are reusable

Basically, a script is a text file containing the statements that comprise a Python program. Once you have created the script, you can execute it over and over without having to retype it each time.

Scripts are editable

Perhaps, more importantly, you can make different versions of the script by modifying the statements from one file to the next using a text editor. Then you can execute each of the individual versions. In this way, it is easy to create different programs with a minimum amount of typing.

You will need a text editor

Just about any text editor will suffice for creating Python script files.

You can use *Microsoft Notepad*, *Microsoft WordPad*, *Microsoft Word*, or just about any word processor if you want to.

Difference between a script and a program

Script:

Scripts are distinct from the core code of the application, which is usually written in a different language, and are often created or at least modified by the end-user. Scripts are often interpreted from source code or byte code, whereas the applications they control are traditionally compiled to native machine code.

Program:

The program has an executable form that the computer can use directly to execute the instructions.

The same program in its human-readable source code form, from which executable programs are derived (e.g., compiled)

Python

What is Python? Chances you are asking yourself this. You may have found this book because you want to learn to program but don't know anything about programming languages. Or you may have heard of programming languages like C, C++, C#, or Java and want to know what Python is and how it compares to "big name" languages. Hopefully I can explain it for you.

Python concepts

If you're not interested in the how's and whys of Python, feel free to skip to the next chapter. In this chapter I will try to explain to the reader why I think Python is one of the best languages available and why it's a great one to start programming with.

- Open-source general-purpose language.
- Object Oriented, Procedural, Functional
- Easy to interface with C/ObjC/Java/Fortran
- Easy-is to interface with C++ (via SWIG)
- Great interactive environment
- Great interactive environment

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 – Python has few keywords, simple structure, and a clearly defined syntax. This allows the student to pick up the language quickly.
- 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-maintained.
- 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.

Dynamic vs. Static

Types Python is a dynamic-typed language. Many other languages are static typed, such as C/C++ and Java. A static typed language requires the programmer to explicitly tell the computer what type of “thing” each data value is.

For example, in C if you had a variable that was to contain the price of something, you would have to declare the variable as a “float” type.

This tells the compiler that the only data that can be used for that variable must be a floating point number, i.e. a number with a decimal point.

If any other data value was assigned to that variable, the compiler would give an error when trying to compile the program.

Python, however, doesn’t require this. You simply give your variables names and assign values to them. The interpreter takes care of keeping track of what kinds of objects your program is using. This also means that you can change the size of the values as you develop the program. Say you have another decimal number (a.k.a. a floating point number) you need in your program. With a static typed language, you have to decide the memory size the variable can take when you first initialize that variable. A double is a floating point value that can handle a much larger number than a normal float (the actual memory sizes depend on the operating environment).

If you declare a variable to be a float but later on assign a value that is too big to it, your program will fail; you will have to go back and change that variable to be a double.

With Python, it doesn't matter. You simply give it whatever number you want and Python will take care of manipulating it as needed. It even works for derived values.

For example, say you are dividing two numbers. One is a floating point number and one is an integer. Python realizes that it's more accurate to keep track of decimals so it automatically calculates the result as a floating point number

Variables

Variables are nothing but reserved memory locations to store values. This means that when you create a variable you reserve some space in memory.

Based on the data type of a variable, the interpreter allocates memory and decides what can be stored in the reserved memory. Therefore, by assigning different data types to variables, you can store integers, decimals or characters in these variables.

Standard Data Types

The data stored in memory can be of many types. For example, a person's age is stored as a numeric value and his or her address is stored as alphanumeric characters. Python has various standard data types that are used to define the operations possible on them and the storage method for each of them.

Python has five standard data types –

- Numbers
- String
- List
- Tuple
- Dictionary

Python Numbers

Number data types store numeric values. Number objects are created when you assign a value to them

Python Strings

Strings in Python are identified as a contiguous set of characters represented in the quotation marks. Python allows for either pairs of single or double quotes. Subsets of strings can be taken using the slice operator ([] and [:]) with indexes starting at 0 in the beginning of the string and working their way from -1 at the end.

Python Lists

Lists are the most versatile of Python's compound data types. A list contains items separated by commas and enclosed within square brackets ([]). To some extent, lists are similar to arrays in C. One difference between them is that all the items belonging to a list can be of different data type. The values stored in a list can be accessed using the slice operator ([] and [:]) with indexes starting at 0 in the beginning of the list and working their way to end -1. The plus (+) sign is the list concatenation operator, and the asterisk (*) is the repetition operator.

Python Tuples

A tuple is another sequence data type that is similar to the list. A tuple consists of a number of values separated by commas. Unlike lists, however, tuples are enclosed within parentheses. The main differences between lists and tuples are: Lists are enclosed in brackets ([]) and their elements and size can be changed, while tuples are enclosed in parentheses (()) and cannot be updated. Tuples can be thought of as read-only lists.

Python Dictionary

Python's dictionaries are kind of hash table type. They work like associative arrays or hashes found in Perl and consist of key-value pairs. A dictionary key can be almost any Python type, but are usually numbers or strings. Values, on the other hand, can be any arbitrary Python object. Dictionaries are enclosed by curly braces ({ }) and values can be assigned and accessed using square braces ([]).

Different modes in python

Python has two basic modes: normal and interactive. The normal mode is the mode where the scripted and finished .py files are run in the Python interpreter. Interactive mode is a command line shell which gives immediate feedback for each statement, while running previously fed statements in active memory. As new lines are fed into the interpreter, the fed program is evaluated both in part and in whole

20 Python libraries

- 1. Requests.** The most famous http library written by Kenneth remits. It's a must have for every python developer.
- 2. Scrappy.** If you are involved in web scraping then this is a must have library for you. After using this library you won't use any other.
- 3. Python.** A guy toolkit for python. I have primarily used it in place of tinder. You will really love it.
- 4. Pillow.** A friendly fork of PIL (Python Imaging Library). It is more user friendly than PIL and is a must have for anyone who works with images.
- 5. SQL Alchemy.** A database library. Many love it and many hate it. The choice is yours.
- 6. Beautiful Soup.** I know it's slow but this xml and html parsing library is very useful for beginners.
- 7. Twisted.** The most important tool for any network application developer. It has a very beautiful ape and is used by a lot of famous python developers.
- 8. Numbly.** How can we leave this very important library? It provides some advance math functionalities to python.
- 9. Skippy.** When we talk about numbly then we have to talk about spicy. It is a library of algorithms and mathematical tools for python and has caused many scientists to switch from ruby to python.
- 10. Matplotlib.** A numerical plotting library. It is very useful for any data scientist or any data analyser.
- 11. Pygmy.** Which developer does not like to play games and develop them? This library will help you achieve your goal of 2d game development.
- 12. Piglet.** A 3d animation and game creation engine. This is the engine in which the famous python port of mine craft was made
- 13. Pit.** A GUI toolkit for python. It is my second choice after python for developing GUI's for my python scripts.
- 14. Pit.** Another python GUI library. It is the same library in which the famous Bit torrent client is created.
- 15. Scaly.** A packet sniffer and analyser for python made in python.



16. Pywin32. A python library which provides some useful methods and classes for interacting with windows.

17. Notch. Natural Language Toolkit – I realize most people won't be using this one, but it's generic enough. It is a very useful library if you want to manipulate strings. But its capacity is beyond that. Do check it out.

18. Nose. A testing framework for python. It is used by millions of python developers. It is a must have if you do test driven development.

19. Simply. Simply can-do algebraic evaluation, differentiation, expansion, complex numbers, etc. It is contained in a pure Python distribution.

20. I Python. I just can't stress enough how useful this tool is. It is a python prompt on steroids. It has completion, history, shell capabilities, and a lot more. Make sure that you take a look at it.

NumPy

NumPy's main object is the homogeneous multidimensional array. It is a table of elements (usually numbers), all of the same type, indexed by a tuple of positive integers. In NumPy dimensions are called *axes*. The number of axes is *rank*.

- Offers Matlab-ish capabilities within Python
- Fast array operations
- 2D arrays, multi-D arrays, linear algebra etc.

Python class and objects

These are the building blocks of OOP. Class creates a new object. This object can be anything, whether an abstract data concept or a model of a physical object, e.g. a chair. Each class has individual characteristics unique to that class, including variables and methods. Classes are very powerful and currently “the big thing” in most programming languages. Hence, there are several chapters dedicated to OOP later in the book.

The class is the most basic component of object-oriented programming. Previously, you learned how to use functions to make your program do something.

Now will move into the big, scary world of Object-Oriented Programming (OOP). To be honest, it took me several months to get a handle on objects.

When I first learned C and C++, I did great; functions just made sense for me.

Having messed around with BASIC in the early '90s, I realized functions were just like subroutines so there wasn't much new to learn.

However, when my C++ course started talking about objects, classes, and all the new features of OOP, my grades definitely suffered.

Once you learn OOP, you'll realize that it's actually a pretty powerful tool. Plus many Python libraries and APIs use classes, so you should at least be able to understand what the code is doing.

One thing to note about Python and OOP: it's not mandatory to use objects in your code in a way that works best; maybe you don't need to have a full-blown class with initialization code and methods to just return a calculation. With Python, you can get as technical as you want.

As you've already seen, Python can do just fine with functions. Unlike languages such as Java, you aren't tied down to a single way of doing things; you can mix functions and classes as necessary in the same program. This lets you build the code

Objects are an encapsulation of variables and functions into a single entity. Objects get their variables and functions from classes. Classes are essentially a template to create your objects.

Here's a brief list of Python OOP ideas:

- The class statement creates a class object and gives it a name. This creates a new namespace.
- Assignments within the class create class attributes. These attributes are accessed by qualifying the name using dot syntax: `ClassName.Attribute`.
- Class attributes export the state of an object and its associated behaviour. These attributes are shared by all instances of a class.
- Calling a class (just like a function) creates a new instance of the class.

This is where the multiple copy's part comes in.

- Each instance gets ("inherits") the default class attributes and gets its own namespace. This prevents instance objects from overlapping and confusing the program.
- Using the term `self` identifies a particular instance, allowing for per-instance attributes. This allows items such as variables to be associated with a particular instance.

Inheritance

First off, classes allow you to modify a program without really making changes to it. To elaborate, by subclassing a class, you can change the behaviour of the program by simply adding new components to it rather than rewriting the existing components. As we've seen, an instance of a class inherits the attributes of that class. However, classes can also inherit attributes from other classes. Hence, a subclass inherits from a superclass allowing you to make a generic



superclass that is specialized via subclasses. The subclasses can override the logic in a superclass, allowing you to change the behaviour of your classes without changing the superclass at all.

Operator Overloads

Operator overloading simply means that objects that you create from classes can respond to actions (operations) that are already defined within Python, such as addition, slicing, printing, etc.

Even though these actions can be implemented via class methods, using overloading ties the behavior closer to Python's object model and the object interfaces are more consistent to Python's built-in objects, hence overloading is easier to learn and use.

User-made classes can override nearly all of Python's built-in operation methods

Exceptions

I've talked about exceptions before but now I will talk about them in depth. Essentially, exceptions are events that modify program's flow, either intentionally or due to errors.

They are special events that can occur due to an error, e.g. trying to open a file that doesn't exist, or when the program reaches a marker, such as the completion of a loop.

Exceptions, by definition, don't occur very often; hence, they are the "exception to the rule" and a special class has been created for them. Exceptions are everywhere in Python.

Virtually every module in the standard Python library uses them, and Python itself will raise them in a lot of different circumstances.

Here are just a few examples:

- Accessing a non-existent dictionary key will raise a Key Error exception.
- Searching a list for a non-existent value will raise a Value Error exception.
- Calling a non-existent method will raise an Attribute Error exception.
- Referencing a non-existent variable will raise a Name Error exception.
- Mixing data types without coercion will raise a Type Error exception.

One use of exceptions is to catch a fault and allow the program to continue working; we have seen this before when we talked about files.

This is the most common way to use exceptions. When programming with the Python command line interpreter, you don't need to worry about catching exceptions.

Your program is usually short enough to not be hurt too much if an exception occurs.

Plus, having the exception occur at the command line is a quick and easy way to tell if your code logic has a problem.

However, if the same error occurred in your real program, it will fail and stop working. Exceptions can be created manually in the code by raising an exception.

It operates exactly as a system-caused exceptions, except that the programmer is doing it on purpose. This can be for a number of reasons. One of the benefits of using exceptions is that, by their nature, they don't put any overhead on the code processing. Because exceptions aren't supposed to happen very often, they aren't processed until they occur. Exceptions can be thought of as a special form of the if/elif statements. You can realistically do the same thing with if blocks as you can with exceptions. However, as already mentioned, exceptions aren't processed until they occur; if blocks are processed all the time. Proper use of exceptions can help the performance of your program. The more infrequent the error might occur, the better off you are to use exceptions; using if blocks requires Python to always test extra conditions before continuing. Exceptions also make code management easier: if your programming logic is mixed in with error-handling if statements, it can be difficult to read, modify, and debug your program.

User-Defined Exceptions

I won't spend too much time talking about this, but Python does allow for a programmer to create his own exceptions. You probably won't have to do this very often but it's nice to have the option when necessary. However, before making your own exceptions, make sure there isn't one of the built-in exceptions that will work for you. They have been "tested by fire" over the years and not only work effectively, they have been optimized for performance and are bug-free. Making your own exceptions involves object-oriented programming, which will be covered in the next chapter. To make a custom exception, the programmer determines which base exception to use as the class to inherit from, e.g. making an exception for negative numbers or one for imaginary numbers would probably fall under the Arithmetic Error exception class. To make a custom exception, simply inherit the base exception and define what it will do.

Python modules

Python allows us to store our code in files (also called modules). This is very useful for more serious programming, where we do not want to retype a long function definition from the very beginning just to change one mistake. In doing this, we are essentially defining our own modules, just like the modules defined already in the Python library.

To support this, Python has a way to put definitions in a file and use them in a script or in an interactive instance of the interpreter. Such a file is called a *module*; definitions from a module can be *imported* into other modules or into the *main* module.

Testing code

As indicated above, code is usually developed in a file using an editor. To test the code, import it into a Python session and try to run it. Usually there is an error, so you go back to the file, make a correction, and test again. This process is repeated until you are satisfied that the code works. This entire process is known as the development cycle. There are two types of errors that you will encounter. Syntax errors occur when the form of some command is invalid. This happens when you make typing errors such as misspellings, or call something by the wrong name, and for many other reasons. Python will always give an error message for a syntax error.

Functions in Python

It is possible, and very useful, to define our own functions in Python. Generally speaking, if you need to do a calculation only once, then use the interpreter. But when you or others have need to perform a certain type of calculation many times, then define a function.

You use functions in programming to bundle a set of instructions that you want to use repeatedly or that, because of their complexity, are better self-contained in a sub-program and called when needed. That means that a function is a piece of code written to carry out a specified task. To carry out that specific task, the function might or might not need multiple inputs. When the task is carved out, the function can or cannot return one or more values.

There are three types of functions in python:

Help (), min (), print ().

Namespaces in Python are implemented as Python dictionaries, this means it is a mapping from names (keys) to objects (values). The user doesn't have to know this to write a Python program and when using namespaces.

Some namespaces in Python:

- global names of a module
- local names in a function or method invocation
- built-in names: this namespace contains built-in functions (e.g. `abs()`, `camp()`, ...) and built-in exception names

Garbage Collection

Garbage Collector exposes the underlying memory management mechanism of Python, the automatic garbage collector. The module includes functions for controlling how the collector operates and to examine the objects known to the system, either pending collection or stuck in reference cycles and unable to be freed.

Tkinter

Tkinter is a popular graphical user interface (GUI) library for creating desktop applications in Python. It is a built-in module in Python, and it provides a simple and easy-to-use interface for creating windows, buttons, labels, text boxes, menus, and other GUI components for desktop applications.

Tkinter is based on the Tcl/Tk GUI toolkit and is available for Windows, macOS, and Linux platforms. It allows developers to create interactive desktop applications with a rich set of GUI components that can be used to create windows, dialogs, buttons, labels, text boxes, check boxes, radio buttons, menus, and more.

Here are some key features of Tkinter:

1. Simple and easy-to-use: Tkinter provides a simple and intuitive interface for creating desktop applications with Python. It has a small learning curve, making it ideal for beginners.
2. Cross-platform: Tkinter is available for Windows, macOS, and Linux platforms, making it a versatile choice for creating desktop applications that can run on multiple operating systems.
3. Customizable widgets: Tkinter provides a rich set of customizable GUI components, including buttons, labels, text boxes, check boxes, radio buttons, menus, and more. These widgets can be easily customized to suit the application's visual appearance and functionality.
4. Event-driven programming: Tkinter follows an event-driven programming paradigm, where the application responds to events such as button clicks, key presses, and mouse actions. This allows for interactive and responsive desktop applications.
5. Integration with Python: Tkinter is a Python library and can be easily integrated with other Python modules and libraries. It allows for the use of Python's features, such as functions, classes, and modules, to create desktop applications with dynamic behavior.
6. Extensive documentation and community support: Tkinter has extensive documentation, including tutorials, examples, and a large community of developers who can provide support and assistance in case of any issues.

Overall, Tkinter is a powerful and widely used GUI library for creating desktop applications in Python. It provides a simple and easy-to-use interface for creating interactive desktop applications with cross-platform compatibility.

Arduino

Arduino is an open-source electronics platform that consists of hardware and software components designed for building interactive projects and prototyping electronic devices. It was initially developed as a simple microcontroller-based board for artists and designers to create interactive installations, but it has since gained widespread popularity among hobbyists, students, educators, and professionals for a wide range of applications.

Arduino hardware typically consists of a microcontroller board that contains various digital and analog input/output pins, which can be used to connect to sensors, actuators, and other electronic components. Arduino boards are available in different sizes, configurations, and capabilities to suit different project requirements. Some of the popular Arduino boards include Arduino Uno, Arduino Mega, Arduino Nano, and Arduino Due, among others.

The Arduino software, also known as the Arduino Integrated Development Environment (IDE), is a user-friendly graphical interface that allows users to write, compile, and upload code to Arduino boards. The Arduino IDE is based on the Processing programming language and provides a simplified programming environment with a rich library of pre-written code and examples for various sensors, actuators, and other peripherals.

One of the key features of Arduino is its open-source nature, which means that the hardware design, software code, and documentation are freely available for anyone to use, modify, and distribute. This has fostered a large and supportive community of Arduino users, who share their projects, knowledge, and resources online through forums, blogs, and other platforms. Arduino has also spawned numerous third-party add-ons, shields, and libraries that extend its capabilities and enable users to interface with a wide range of sensors, actuators, and communication protocols.

Arduino is used in a wide range of applications, including home automation, robotics, Internet of Things (IoT) devices, wearable technology, environmental monitoring, scientific research, art installations, and many more. It is known for its ease of use, affordability, and versatility, making it a popular choice for both beginners and experienced users in the field of electronics and programming.

GSM Modem

A GSM modem, also known as a GSM module or GSM modem module, is a device that allows communication between a computer or microcontroller and a GSM (Global System for Mobile Communications) network. It enables sending and receiving SMS (Short Message Service) text messages, making and receiving voice calls, and accessing data services over the GSM network.

GSM modems are typically small electronic devices that incorporate a SIM (Subscriber Identity Module) card slot to provide authentication and access to the GSM network. They are designed to be used in a variety of applications, such as remote monitoring and control, mobile payment systems, vehicle tracking, smart agriculture, and other Internet of Things (IoT) applications where wireless communication is required.

GSM modems typically connect to a computer or microcontroller via a serial interface, such as RS232 or USB, and use AT (Attention) commands to communicate with the device. AT commands are a standardized set of commands that can be used to control various functions of the GSM modem, such as sending and receiving SMS messages, making voice calls, and configuring network settings.

One of the key features of GSM modems is their ability to operate on different GSM frequency bands, which allows them to be used in various regions and countries around the world. GSM networks operate on different frequency bands, and the modem must support the appropriate frequency bands for the desired region or country.

GSM modems are widely used in many applications that require wireless communication capabilities, especially in areas where cellular networks are available and reliable. They provide a convenient and cost-effective way to add wireless communication functionality to a wide range of devices and systems, making them a popular choice for IoT applications, remote monitoring and control, and other wireless communication needs.



CHAPTER – 8

IMPLEMENTATION

8.1. MODULES:

1.Demographic data: Demographic analysis is the study of a population-based on factors such as age, race, and sex. Demographic data refers to socioeconomic information expressed statistically, including education. A higher education institute aims to provide a quality education to the students for achieving outstanding performance on their part. Students' academic performance is the most important quality measure that depends on several factors such as demographics, personality traits, socio-economic, and other environmental factors. The knowledge about these factors and their effect on students' performance can assist in managing their impact. This module involves the collection of demo graphical data of each student.

2.Academic history: Academic performance/ achievement is the extent to which a student, teacher, or institution has attained their short or long-term educational goals and is measured either by continuous assessment or cumulative grade point average.so here needed to collect the previous academic data of student there by introduce a model that is trained with data and test the new data.

3.Attendance and engagement: In general, regular class attendance has the strongest positive correlation with course grade, stronger than motivation, high school GPA and conscientiousness. Class attendance is especially important in courses with technical and practical subject matter.

4.Behavioral data: When working with data, it is critical not just the volume and quality of data but also their relevance, that is, the ability of such data to provide information about the characteristics that are wanted to be studied. The problem is that often these data are not available and we use instead less relevant but more available data Within the world of education, Learning Management Systems (LMSs) provide a large amount of data from the student activity. However, these data reflect the use of the platform but they do not describe the results of the learning process of students, that is, they describe the behaviour but not the learning results.

5.Family background: Ibrahima Naite (2021) found that students with highly involved parents had better academic performance and higher test scores in all the subjects compared to students

whose parents were not involved in their education. The importance of family education on a student's education cannot be overstated. Family members have a vital part in their children's education. The method in which a youngster is immersed in a learning environment is determined by his or her family history. A child's academic success also influences his familial background.

6.TKinter: Tkinter is a popular graphical user interface (GUI) library for creating desktop applications in Python. It is a built-in module in Python, and it provides a simple and easy-to-use interface for creating windows, buttons, labels, text boxes, menus, and other GUI components for desktop applications.

7.GSM Modem: A GSM modem, also known as a GSM module or GSM modem module, is a device that allows communication between a computer or microcontroller and a GSM (Global System for Mobile Communications) network. It enables sending and receiving SMS (Short Message Service) text messages, making and receiving voice calls, and accessing data services over the GSM network.

CHAPTER – 9

TESTING

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 test. Each test type addresses a specific testing requirement.

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 satisfactory, 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.

SYSTEMTEST

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 the software tester has knowledge of the inner workings, structure and language of the software, or at least its 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 strategy and approach Field testing will be performed manually and functional tests will be written in detail.

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.

The task of the integration test is to check that components or software applications, e.g. components in a software system or – one step up – software applications at the company level – interact without error.

Test Results: All the test cases mentioned above passed successfully. No defects encountered.

6.3 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.

Test Results: All the test cases mentioned above passed successfully. No defects encountered.

CHAPTER – 10

RESULTS

OUTPUT SCREEN SHOTS WITH DESCRIPTION.

GUI:

Here user view the GUI interface

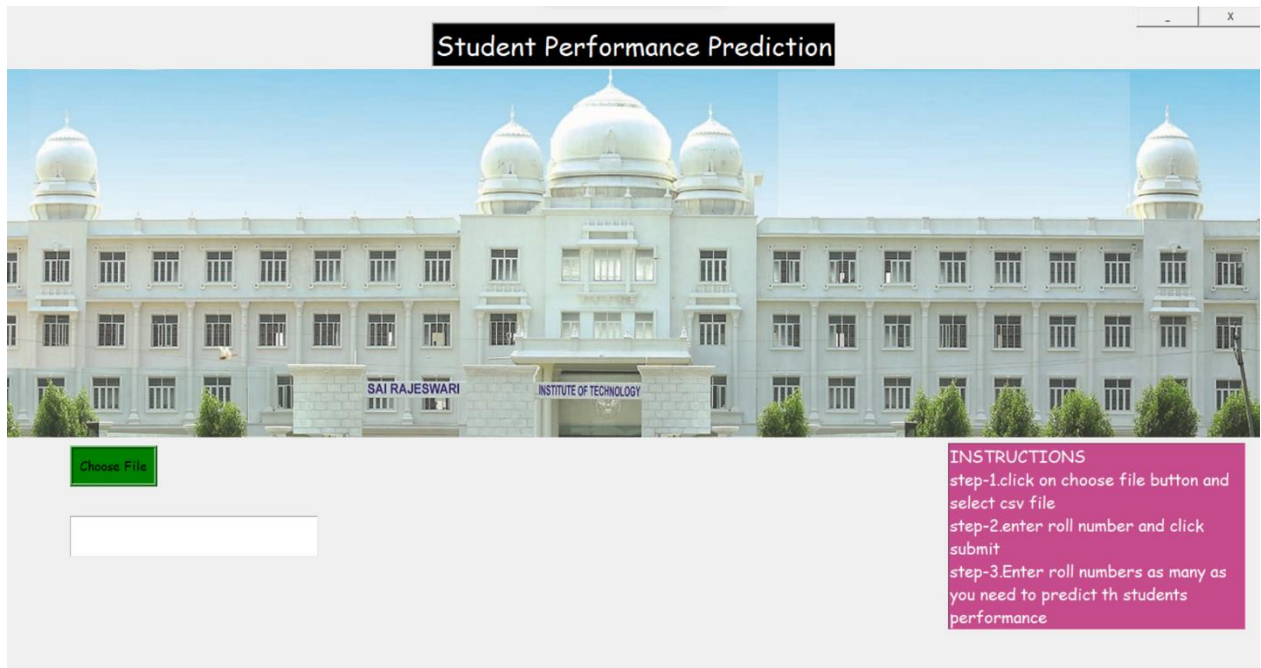


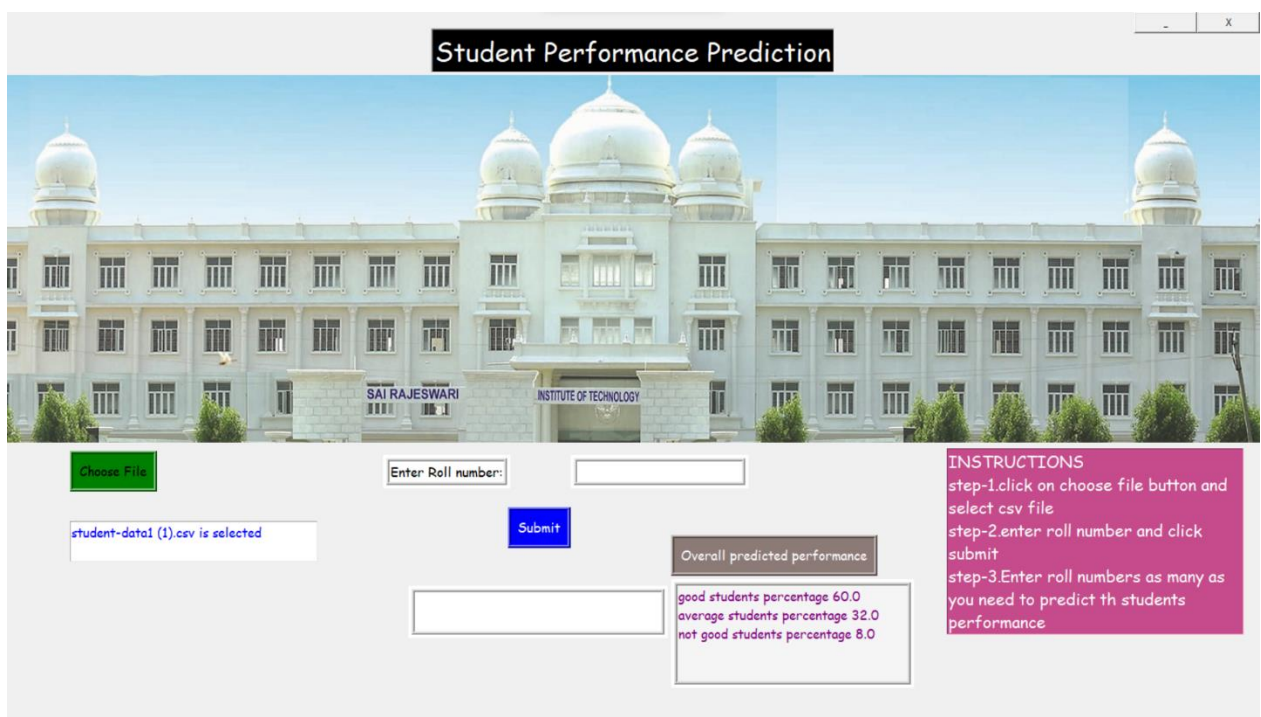
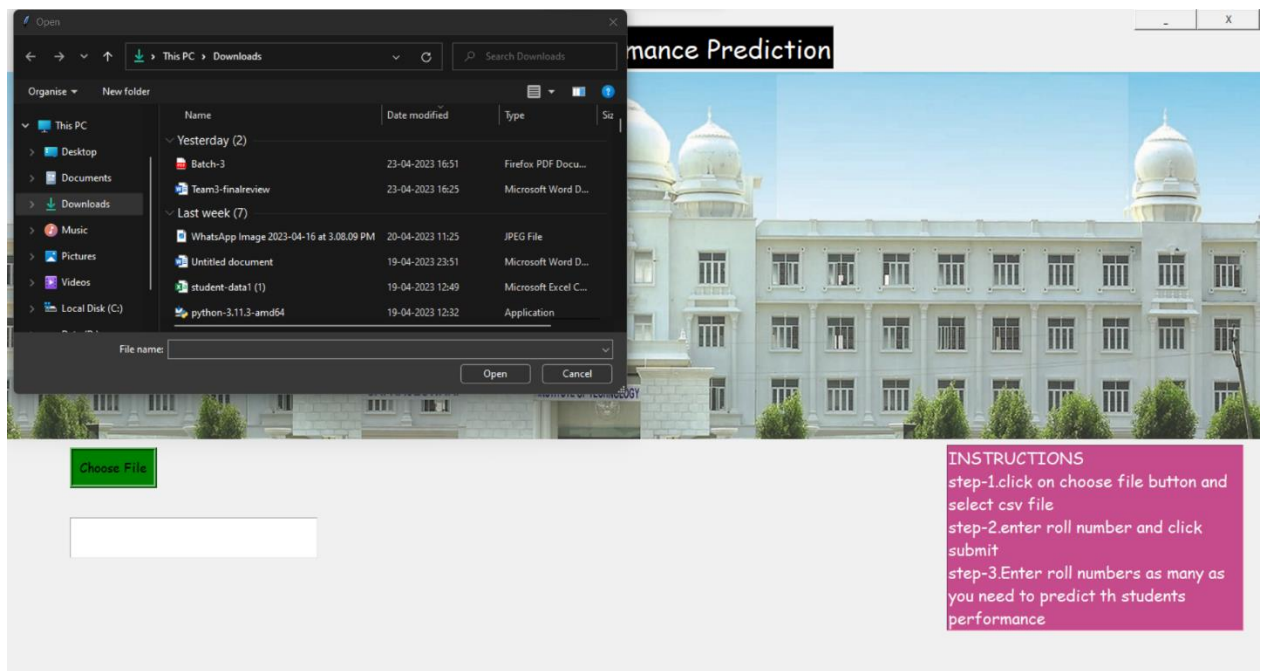
Fig1:GUI interface

Choose file:

This is a button where we can able to upload CSV file which contains student database.

Whenever user click on the choose file button it opens the window contains all the documents available in the hard disk. The below figure shows the output screenshot after click the choose file button.


After selecting the file few more buttons are visible where we can able to search student roll number inorder by getting the advanced prediction of student performance.



The above output screenshot contains the some fields such as Enter Roll number, submit etc. where we can enter the roll number on the empty field and click on the submit button there by we get the student performance prediction. There is one more field where it contains the good

students percentage, average students percentage and not good students percentage from overall students.

Student Performance Prediction



Choose File

student-data1 (1).csv is selected

Enter Roll number: 198R1A0538


Submit

GOOD-PASSED

Overall predicted performance
good students percentage 60.0
average students percentage 32.0
not good students percentage 8.0

INSTRUCTIONS
step-1.click on choose file button and select csv file
step-2.enter roll number and click submit
step-3.Enter roll numbers as many as you need to predict th students performance

Student Performance Prediction



Choose File

student-data1 (1).csv is selected

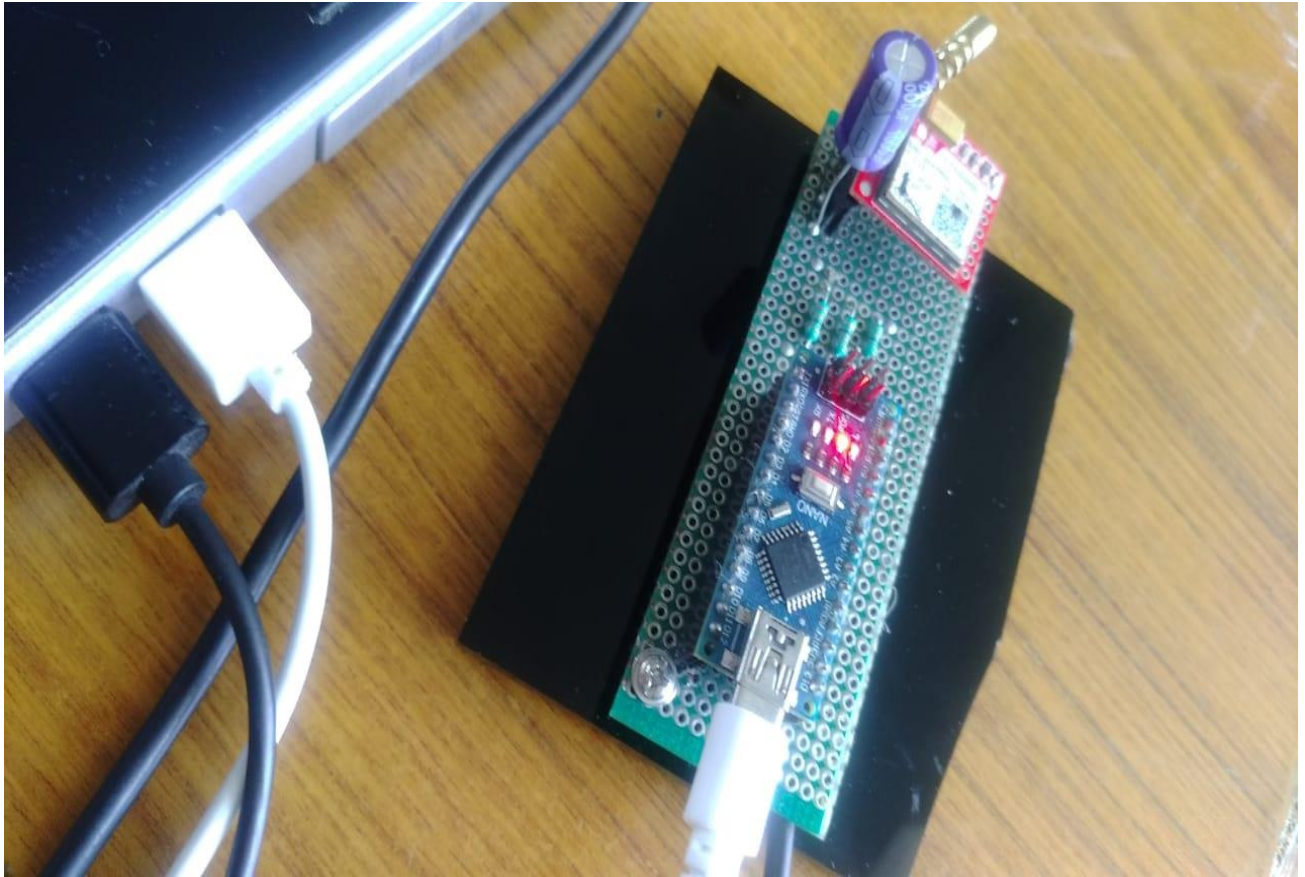
Enter Roll number: 198R1A0501

Submit

Average-Passed

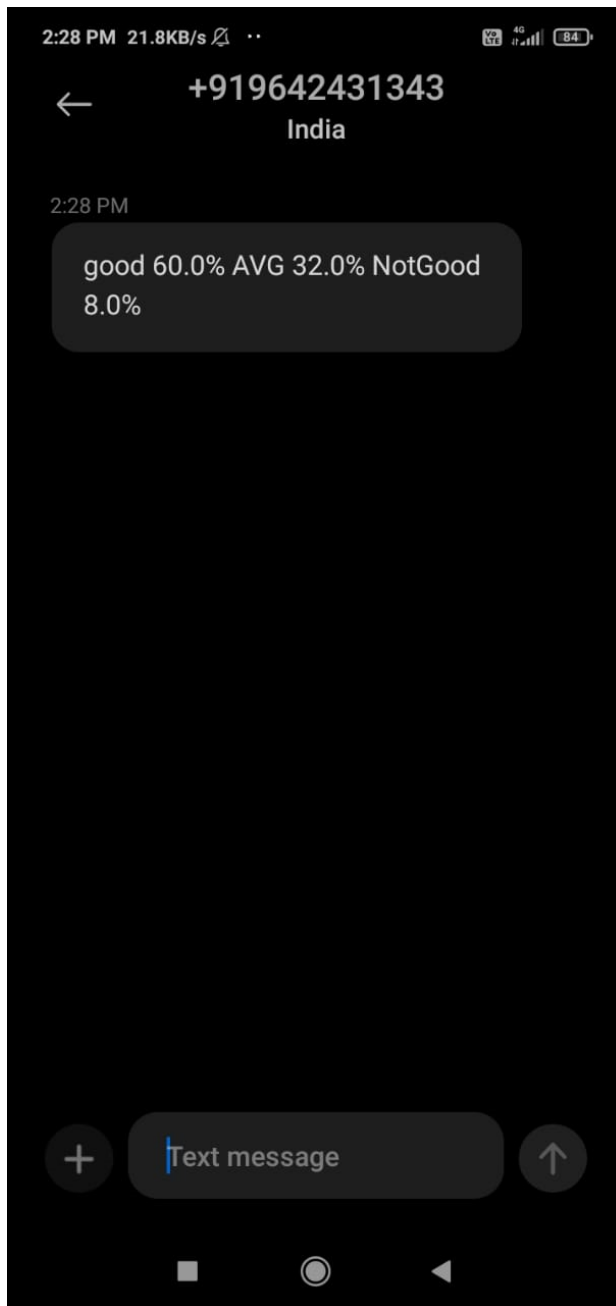
Overall predicted performance
good students percentage 60.0
average students percentage 32.0
not good students percentage 8.0

INSTRUCTIONS
step-1.click on choose file button and select csv file
step-2.enter roll number and click submit
step-3.Enter roll numbers as many as you need to predict th students performance



The above screenshot screen shot shows the Arduino kit with GSM Modem these are connected to the PC and also we need to dump the code to the Arduino board and also need to give the power supply through PC or battery there by GSM will works. GSM Modem can able to send the message notifications to respective teacher.

The below screenshot shows the message notification received by mentor/teacher from the GSM modem where the message contains overall performance of the particular class.



CHAPTER – 11

CONCLUSION

Improving the educational system is a big deal, As student engineers, we can contribute to realizing this goal by using technologies and study resources like machine learning materials ,to come up with an innovative solution to help the student in need especially student who live in hard conditions (demographic, social and educational problems). In this project, we come up with the idea to build a model that predicts student status based on different features. Our main challenges were to define the best classification algorithm and to identify the most impactful factors for student academic status, to provide them with a summary or valedictorian of student's best conditions to reach high academics status and avoid failure. For this project entitled "Students-performance and difficulties prediction"", we used several methods for classification such as logistic regression, KNN, and SVM and we evaluate this model using different metrics like f1 score, roc curve, and confusion matrix and finally we got a winner which SVM with an accuracy of 84% compared to KNN and logestic regression. Before achieving our main challenges they were several steps to take: -data processing -data visualization -Models implementation -comparison of 3 algorithm

FUTURE SCOPE

There are quite a few things that can be polished or be added in the future work. showing the all percentages of students in pictorial form but that will not possible in python IDLE. The result in pictorial representation gives more clarity to the mentors by just saw that graphs. And also adding one more algorithm may gives more accuracy.

BIBLIOGRAPHY

SOFTWARE INSTALLATION FOR MACHINE LEARNING PROJECTS

Installing Python:

1. To download and install Python visit the official website of Python <https://www.python.org/downloads/> and choose your version.



2. Once the download is complete, run the exe for install Python. Now click on Install Now.
3. You can see Python installing at this point.
4. When it finishes, you can see a screen that says the Setup was successful. Now click on "Close".

1. You need to install some packages to execute your project in a proper way.
2. Open the command prompt/ anaconda prompt or terminal as administrator.
- 3 The prompt will get open, with specified path, type “pip install package name” which you want to install (like NumPy, pandas, sea born, scikit-learn, Matplotlib, Pyplot)

Ex: Pip install NumPy

```
C:\WINDOWS\system32>pip install numpy==1.18.5
Collecting numpy==1.18.5
  Downloading numpy-1.18.5-cp36-cp36m-win_amd64.whl (12.7 MB)
    |████████████████████████████████████████| 12.7 MB 939 kB/s
ERROR: tensorboard 2.0.2 has requirement setuptools>=41.0.0, b
Installing collected packages: numpy
Successfully installed numpy-1.18.5
```

REFERENCES

- [1] Chuang Liu & Haojie Wang & Zhonghu Yuan, 2022. "[A Method for Predicting the Academic Performances of College Students Based on Education System Data](#)," [Mathematics](#), MDPI, vol. 10(20), pages 1-19, October.
- [2] Dr. A.P. Siva Kumar and Chidananda K, Advanced Prediction of a student in a university using Machine Learning techniques, Vol. 12 No. 14 (2021): Vol. 12 No. 14 (2021)
- [3]Havan Agrawal, Harshil Mavani ,Student Performance Prediction using Machine Learning , Volume 13, Number 2 (2018) pp. 1171-1176
- [4]. Safira Begum, Sunita S Padmannavar, "Prediction of Student Performance using Genetically Optimized Feature Selection with Multiclass Classification," *International Journal of Engineering Trends and Technology*, vol. 70, no. 4, pp. 223-235, 2022.
- [5]. Toward Predicting Student's Academic Performance Using Artificial Neural Networks (ANNs) ,YGA, The Eurasia Proceedings of Educational & Social Sciences (EPESS), 2014 Volume 1, Pages 68-72
- [6].Comparative Analysis of Various Techniques used for Predicting Student's Performance Amita Dhankhara , Kamna Solankib
- [7]. Student's Performance: A Systematic Literature Review Lidia Sandra , Ford Lumbangaol , Tokuro Matsuo, **TEM Journal, Vol.10, No.4, November 2021**
- [8]. Predicting Students' Performance Using Machine Learning Techniques Hussein Altabrawee a osama Abdul Jaleel Ali a , Samir Qaisar Ajmi, Journal of University of Babylon, Pure and Applied Sciences, Vol.(27), No.(1): 2019