

**ST. JOHN'S SENIOR SECONDARY SCHOOL & JR
COLLEGE**

MANDAVELI, CHENNAI-28



ARTIFICIAL INTELLIGENCE PROJECT

[2024-2025]

TITLE: STUDENTS RESULTS PREDICTOR

SUBMITTED BY: MUKESHWAR RAUDRA.J

TEAM MEMBERS:

**MUKESHWAR RAUDRA.J
MOHAMMED RAYAAN.B
CHARANJET NATHAN.S
DEGA YESWANTH
SAMEER AHMED.S**



ST. JOHN'S SENIOR SECONDARY SCHOOL & JR COLLEGE

MANDAVELI, CHENNAI-28

CERTIFICATE

This is to certify that **MUKESHWAR RAUDRA.J** , of Std. XII Reg. No_____, of St. John's Senior Secondary School, has successfully completed the project in **ARTIFICIAL INTELLIGENCE** under the guidance of **MR. VIMAL RAJ B.** This project is based on the curriculum and meant for CBSE Delhi 2024-25 SSCE Practical Examination, and the student has provided an excellent account of it.

SIGNATURE OF THE
INTERNAL EXAMINER

SIGNATURE OF THE
SENIOR PRINCIPAL

SIGNATURE OF THE
EXTERNAL EXAMINER

ACKNOWLEDGEMENT

I would like to express my deepest gratitude to my project supervisor, **MR. VIMAL RAJ B., PGT in AI**, for his invaluable guidance and support. I also extend my gratitude to the senior principal and correspondent, **DR. NALINI WILSON**, at **ST. JOHN'S SENIOR SECONDARY SCHOOL & JUNIOR COLLEGE**, for her assistance in accessing research materials. I am grateful to all those who helped me in the completion of the project. Additionally, I extend my appreciation to my classmates for their constructive feedback and encouragement.

Yours Sincerely,
MUKESHWAR RAUDRA.J

INDEX

1.INTRODUCTION

1.1	OVERVIEW OF STUDENTS RESULT PREDICTOR(SRP)	2
1.2	AIM OF THE PROJECT	3

3. DESIGN AND DEVELOPMENT OF THE SYSTEM

3.1	CODE	4-10
3.2	PYTHON OUTPUT	11-15
3.3	MYSOL OUTPUT	16-19
3.5	EXPLANATION ABOUT THE PROGRAMME	20-26
3.6	BASIC NEEDS	27-28

4. GENERAL POINTS OF STUDENTS RESULT PREDICTOR

4.1	BENEFITS OF USING STUDENTS RESULT PREDICTOR	29-32
4.3	CONCLUSION	33

5. BIBLIOGRAPHY	34
-----------------------	----

1.Introduction to SRP



1.1 OVERVIEW OF STUDENTS RESULT PREDICTOR:

A student results predictor is a tool or system designed to estimate a student's academic performance based on various inputs. It uses data such as past grades, attendance records, study habits, and sometimes socio-economic factors to predict future results. These predictors can leverage statistical models, machine learning, or artificial intelligence to provide insights on likely outcomes, helping educators, students, and parents identify areas for improvement and make informed decisions. Predictors can vary in complexity, from simple regression models to more advanced neural networks, and can be applied across different educational levels. These predictors use statistical models, machine learning algorithms, or data analytics to forecast likely grades or academic achievements, helping educators and students identify strengths, weaknesses, and areas for improvement. By analyzing patterns in the data, student results predictors can aid in personalizing educational support, setting realistic goals, and enhancing academic planning for better student outcomes..

1.2 AIM OF THE PROJECT:

A student results predictor is a tool or system designed to estimate or predict students' academic outcomes based on various factors such as past grades, attendance, participation, study habits, and other performance indicators. These predictors use statistical models, machine learning algorithms, or data analytics to forecast likely grades or academic achievements, helping educators and students identify strengths, weaknesses, and areas for improvement. By analyzing patterns in the data, student results predictors can aid in personalizing educational support, setting realistic goals, and enhancing academic planning for better student outcomes



the Student Results Predictor project is to develop a reliable tool that accurately forecasts students' academic outcomes based on historical data and performance indicators. This tool will use data-driven models to help students, educators, and institutions make informed decisions, enabling early intervention, personalized academic support, and improved overall performance. By identifying at-risk students and highlighting potential achievers, the project seeks to enhance educational outcomes and contribute to more targeted and effective learning strategies.

CODE

```
1 import pandas as pd
2 import mysql.connector
3 from sqlalchemy import create_engine, text
4 from sklearn.model_selection import train_test_split
5 from sklearn.linear_model import LinearRegression
6 from sklearn.metrics import mean_squared_error, r2_score
7 from sklearn.exceptions import UndefinedMetricWarning
8 from tabulate import tabulate
9 import warnings
10
11 # Database configuration
12 DB_HOST = "localhost"
13 DB_USER = "root"
14 DB_PASSWORD = "NIKOLATESLA369"
15 DB_NAME = "school_db"
16
17 def create_database():
18     # Connect without specifying the database to create it if it doesn't exist
19     connection = mysql.connector.connect(
20         host=DB_HOST,
21         user=DB_USER,
22         password=DB_PASSWORD
23     )
24     cursor = connection.cursor()
25
26     # Create database if it doesn't exist
27     cursor.execute("CREATE DATABASE IF NOT EXISTS school_db")
28     print("Database 'school_db' created or verified successfully.")
29     connection.close()
30
```



```

31 def create_student_table():
32     # Ensure database is created
33     create_database()
34
35     # Connect to 'school_db' database
36     engine = create_engine(f"mysql+mysqlconnector://{DB_USER}:{DB_PASSWORD}@{DB_HOST}/{DB_NAME}")
37
38     # Define the SQL command to create the students table
39     create_table_query = """
40     CREATE TABLE IF NOT EXISTS students (
41         Roll_Number INT PRIMARY KEY AUTO_INCREMENT,
42         name VARCHAR(100),
43         attendance INT,
44         hours_studied FLOAT,
45         weekly_study_hours FLOAT,
46         previous_score FLOAT,
47         assignments_completed INT,
48         stress_level INT,
49         learning_style VARCHAR(20),
50         extracurriculars_involved INT,
51         goal_score FLOAT,
52         score FLOAT
53     )
54     """
55
56     # Execute the table creation query
57     with engine.connect() as connection:
58         connection.execute(text(create_table_query))
59         print("Table 'students' created or verified successfully.")
60
61 def create_marksheet_table():
62     # Connect to MySQL and create the MARKSHEET table if it doesn't exist
63     connection = mysql.connector.connect(host=DB_HOST, user=DB_USER, password=DB_PASSWORD, database=DB_NAME)
64     cursor = connection.cursor()
65     create_table_query = """
66     CREATE TABLE IF NOT EXISTS MARKSHEET (
67         Roll_Number INT PRIMARY KEY AUTO_INCREMENT,
68         Name VARCHAR(100),
69         Stream VARCHAR(50),
70         Physics FLOAT,
71         Maths FLOAT,
72         Chemistry FLOAT,
73         English FLOAT,
74         Computer_Science FLOAT,
75         Artificial_Intelligence FLOAT,
76         Biology FLOAT,
77         Accountancy FLOAT,
78         Economics FLOAT,
79         Business_Studies FLOAT,
80         IP FLOAT,
81         Total_Marks FLOAT,
82         Percentage FLOAT,
83         Result VARCHAR(10)
84     )
85     """
86     cursor.execute(create_table_query)
87     connection.close()
88     print("Table 'MARKSHEET' created or verified successfully.")
89

```



```

90 def create_db_connection():
91     engine = create_engine(f"mysql+mysqlconnector://{DB_USER}:{DB_PASSWORD}@{DB_HOST}/{DB_NAME}")
92     return engine
93
94 # Display 'students' table data using tabulate
95 def display_table():
96     engine = create_db_connection()
97     query = "SELECT * FROM students"
98     data = pd.read_sql(query, engine)
99     if data.empty:
100         print("No data in 'students' table.")
101     else:
102         print("\nCurrent 'students' Table Data:")
103         print(tabulate(data, headers="keys", tablefmt="fancy_grid"))
104
105 # Search for a student by roll number
106 def search_student_by_roll(Roll_Number):
107     engine = create_db_connection()
108     query = f"SELECT * FROM students WHERE Roll_Number = {Roll_Number}"
109     data = pd.read_sql(query, engine)
110     if data.empty:
111         print(f"No student found with Roll Number: {Roll_Number}")
112     else:
113         print("\nStudent Details:")
114         print(tabulate(data, headers="keys", tablefmt="fancy_grid"))
115
116 # Fetch student data from MySQL database for training
117 def fetch_student_data():
118     engine = create_db_connection()
119     query = """
120     SELECT attendance, hours_studied, weekly_study_hours, previous_score, assignments_completed,
121            stress_level, learning_style, extracurriculars_involved, goal_score, score
122     FROM students
123     """
124     data = pd.read_sql(query, engine)
125     return data
126
127 # Insert a student's data into the MySQL database
128 def insert_student_data(name, attendance, hours_studied, weekly_study_hours, previous_score, assignments_completed, stress_level, learning_style, extracurriculars_involved, goal_score, score):
129     engine = create_db_connection()
130     try:
131         with engine.begin() as connection:
132             insert_query = text("""
133             INSERT INTO students (name, attendance, hours_studied, weekly_study_hours, previous_score, assignments_completed, stress_level, learning_style, extracurriculars_involved, goal_score, score)
134             VALUES (:name, :attendance, :hours_studied, :weekly_study_hours, :previous_score, :assignments_completed, :stress_level, :learning_style, :extracurriculars_involved, :goal_score, :score)
135             """)
136

```

```
137         # Execute the insert statement
138         connection.execute(insert_query, {
139             "name": name,
140             "attendance": attendance,
141             "hours_studied": hours_studied,
142             "weekly_study_hours": weekly_study_hours,
143             "previous_score": previous_score,
144             "assignments_completed": assignments_completed,
145             "stress_level": stress_level,
146             "learning_style": learning_style,
147             "extracurriculars_involved": extracurriculars_involved,
148             "goal_score": goal_score,
149             "score": score
150         })
151         print("Student data inserted successfully.")
152     except Exception as e:
153         print(f"Failed to insert data: {e}")
154
155 # Function to get data from the user and store it in the database
156 def get_student_data():
157     print("Enter student details:")
158     name = input("Name: ")
159     attendance = int(input("Attendance (as a percentage): "))
160     hours_studied = float(input("Hours Studied: "))
161     weekly_study_hours = float(input("Weekly Study Hours: "))
162     previous_score = float(input("Previous Score out of 500: "))
163     assignments_completed = int(input("Assignments Completed (1-30): "))
164     stress_level = int(input("Stress Level (1-10): "))
165     learning_style = input("Learning Style (Visual/Auditory/Kinesthetic): ")
166     extracurriculars_involved = int(input("Extracurricular Involvement (number of activities): "))
167     goal_score = float(input("Goal Score for the term out of 500: "))
168     score = float(input("Current Exam Score out of 500: "))
169
170     insert_student_data(name, attendance, hours_studied, weekly_study_hours,
171     previous_score, assignments_completed, stress_level, learning_style,
172     extracurriculars_involved, goal_score, score)
```



```

172 # Early Warning System: Identify students at risk of underperforming
173 def early_warning_system(predicted_score, threshold=50):
174     if predicted_score < threshold:
175         print("Warning: Student at risk of underperforming!")
176         print("Sending notification to parents and teachers...")
177
178 # Train the model and predict based on user data
179 def train_and_predict():
180     df = fetch_student_data()
181     if df.empty:
182         print("No data available. Please add data before training the model.")
183     return
184
185 # Encode learning styles and drop the original column
186 df["learning_style_encoded"] = df["learning_style"].map({"Visual": 1, "Auditory": 2, "Kinesthetic": 3})
187 df.drop(columns=["learning_style"], inplace=True)
188
189 # Define features (X) and target (y)
190 X = df.drop(columns=["score"])
191 y = df["score"]
192
193 if len(df) < 2:
194     print("Not enough data to split. Please add more student data.")
195     return
196
197 # Split data into training and testing sets
198 X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state=0)
199 model = LinearRegression()
200 model.fit(X_train, y_train)
201
202 # Predict on test data
203 y_pred = model.predict(X_test)
204
205 # Evaluate the model
206 with warnings.catch_warnings():
207     warnings.simplefilter("ignore", category=UndefinedMetricWarning)
208     mse = mean_squared_error(y_test, y_pred)
209     r2 = r2_score(y_test, y_pred)
210
211 print("Model trained successfully.")
212 print("Mean Squared Error:", mse)
213 print("R-squared Score:", r2 if len(y_test) > 1 else "N/A - Not enough data")
214
215 # Ensure feature names match between training and prediction
216 print("\nEnter details for a new student to predict their score.")
217
218 # Create new student data with the correct order of columns
219 new_student = pd.DataFrame({
220     "attendance": [int(input("Attendance (as a percentage): "))],
221     "hours_studied": [float(input("Hours Studied: "))],
222     "weekly_study_hours": [float(input("Weekly Study Hours: "))],
223     "previous_score": [float(input("Previous Score out of 500: "))],
224     "assignments_completed": [int(input("Assignments Completed (1-30): "))],
225     "stress_level": [int(input("Stress Level (1-10): "))],
226     "learning_style_encoded": [int(input("Learning Style (1=Visual, 2=Auditory, 3=Kinesthetic): "))],
227     "extracurriculars_involved": [int(input("Extracurricular Involvement (number of activities): "))],
228     "goal_score": [float(input("Goal Score for the term out of 500: "))]
229 })
230
231 # Ensure the new_student DataFrame has the same columns as X

```



```

282         cursor.execute(insert_marksheet_query, (
283             Roll_Number, name, "Science", scores["Physics"], scores["Maths"], scores["Chemistry"],
284             scores["English"], scores["Computer_Science"], scores["Artificial_Intelligence"],
285             scores["Biology"], scores["Accountancy"], scores["Economics"], scores["Business_Studies"],
286             scores["IP"], Total_Marks, Percentage, Result
287         ))
288
289     connection.commit()
290     cursor.close()
291     connection.close()
292     print("Marksheet created successfully.")
293
294 # Main function to run the program
295 if __name__ == "__main__":
296     create_student_table()
297     create_marksheet_table()
298
299     while True:
300         print("\nOptions:")
301         print("1. Add Student Data")
302         print("2. Display Students Table")
303         print("3. Search Student by Roll Number")
304         print("4. Train Model and Predict Score")
305         print("5. Create Marksheet")
306         print("6. Exit")
307
308         choice = input("Select an option (1-6): ")
309
310         if choice == "1":
311             get_student_data()
312         elif choice == "2":
313             display_table()
314         elif choice == "3":
315             roll_number = int(input("Enter Roll Number to search: "))
316             search_student_by_roll(roll_number)
317         elif choice == "4":
318             train_and_predict()
319         elif choice == "5":
320             create_marksheet()
321         elif choice == "6":
322             print("Exiting the program.")
323             break
324         else:
325             print("Invalid choice. Please select a valid option.")
326

```

*** CODE ENDED ***

3.2 PYTHON OUTPUT

Python 3.12.7 (tags/v3.12.7:0b05ead, Oct 1 2024, 03:06:41) [MSC v.1941 64 bit (AMD64)] on win32
Type "help", "copyright", "credits" or "license()" for more information.

```
===== RESTART: C:\Mukesh Project\A.I PROJECT\OG.py =====  
Database 'school_db' created or verified successfully.  
Table 'students' created or verified successfully.  
Table 'MARKSHEET' created or verified successfully.
```

Options:

1. Add Student Data
2. Display Students Table
3. Search Student by Roll Number
4. Train Model and Predict Score
5. Create Marksheet
6. Exit

Select an option (1-6): 1

Enter student details:

Name: MERCURY

Attendance (as a percentage): 80

Hours Studied: 7.8

Weekly Study Hours: 7.5

Previous Score out of 500: 413

Assignments Completed (1-30): 28

Stress Level (1-10): 7

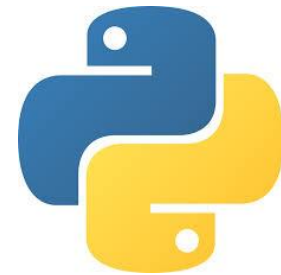
Learning Style (Visual/Auditory/Kinesthetic): Kinesthetic

Extracurricular Involvement (number of activities): 16

Goal Score for the term out of 500: 490

Current Exam Score out of 500: 470

Student data inserted successfully.



Options:

1. Add Student Data
2. Display Students Table
3. Search Student by Roll Number
4. Train Model and Predict Score
5. Create Marksheet
6. Exit

Select an option (1-6): 1

Enter student details:

Name: GANDHI

Attendance (as a percentage): 90

Hours Studied: 9.8

Weekly Study Hours: 8.7

Previous Score out of 500: 470.6

Assignments Completed (1-30): 30

Stress Level (1-10): 3

Learning Style (Visual/Auditory/Kinesthetic): Visual

Extracurricular Involvement (number of activities): 24

Goal Score for the term out of 500: 500.0

Current Exam Score out of 500: 498.4

Student data inserted successfully.

Options:

1. Add Student Data
2. Display Students Table
3. Search Student by Roll Number
4. Train Model and Predict Score
5. Create Marksheet
6. Exit

Select an option (1-6): 1

Enter student details:

Name: RADIUM

Attendance (as a percentage): 75

Hours Studied: 8.6

Weekly Study Hours: 6.6

Previous Score out of 500: 413

Assignments Completed (1-30): 25

Stress Level (1-10): 7

Learning Style (Visual/Auditory/Kinesthetic): Visual

Extracurricular Involvement (number of activities): 10

Goal Score for the term out of 500: 499.8

Current Exam Score out of 500: 453.4

Student data inserted successfully.

Options:

1. Add Student Data
2. Display Students Table
3. Search Student by Roll Number
4. Train Model and Predict Score
5. Create Marksheet
6. Exit

Select an option (1-6): 1

Enter student details:

Name: SANJAY

Attendance (as a percentage): 50

Hours Studied: 10.8

Weekly Study Hours: 9.9

Previous Score out of 500: 498.9

Assignments Completed (1-30): 30

Stress Level (1-10): 10

Learning Style (Visual/Auditory/Kinesthetic): Kinesthetic

Extracurricular Involvement (number of activities): 30

Goal Score for the term out of 500: 500.0

Current Exam Score out of 500: 499.0

Student data inserted successfully.

Options:

1. Add Student Data
2. Display Students Table
3. Search Student by Roll Number
4. Train Model and Predict Score
5. Create Marksheet
6. Exit

Select an option (1-6): 2

Current 'students' Table Data:

	Roll_Number	name	attendance	hours_studied	weekly_study_hours	previous_score	assignments_completed	stress_level	learning_style	extracurriculars_involved	goal_score	score
0	1	MERCURY	80	7.8	7.5	413	28	7	Kinesthetic	16	490	470
1	2	GANDHI	90	9.8	8.7	470.6	30	3	Visual	24	500	498.4
2	3	RADIUM	75	8.6	6.6	413	25	7	Visual	10	499.8	453.4
3	4	SANJAY	50	10.8	9.9	498.9	30	10	Kinesthetic	30	500	499

Options:

1. Add Student Data
2. Display Students Table
3. Search Student by Roll Number
4. Train Model and Predict Score
5. Create Marksheet
6. Exit

Select an option (1-6): 3

Enter Roll Number to search: 3

Student Details:

	Roll_Number	name	attendance	hours_studied	weekly_study_hours	previous_score	assignments_completed	stress_level	learning_style	extracurriculars_involved	goal_score	score
0	3	RADHUM	75	8.6	6.6	413	25	7	Visual	10	499.8	453.4

Options:

1. Add Student Data
2. Display Students Table
3. Search Student by Roll Number
4. Train Model and Predict Score
5. Create Marksheet
6. Exit

Select an option (1-6): 4

Model trained successfully.

Mean Squared Error: 2026.7478997268695

R-squared Score: N/A - Not enough data

Enter details for a new student to predict their score.

Attendance (as a percentage): 82

Hours Studied: 9.2

Weekly Study Hours: 10.0

Previous Score out of 500: 450

Assignments Completed (1-30): 29

Stress Level (1-10): 9

Learning Style (1=Visual, 2=Auditory, 3=Kinesthetic): 1

Extracurricular Involvement (number of activities): 12

Goal Score for the term out of 500: 500

Predicted Score: 497.31

Options:

1. Add Student Data
2. Display Students Table
3. Search Student by Roll Number
4. Train Model and Predict Score
5. Create Marksheet
6. Exit

Select an option (1-6): 5

Marksheet created successfully.

Options:

1. Add Student Data
2. Display Students Table
3. Search Student by Roll Number
4. Train Model and Predict Score
5. Create Marksheet
6. Exit

Select an option (1-6): 6

Exiting the program.

3.3 MYSQL OUTPUT

```
Enter password: *****
Welcome to the MySQL monitor.  Commands end with ; or \g.
Your MySQL connection id is 121
Server version: 8.0.20 MySQL Community Server - GPL

Copyright (c) 2000, 2020, Oracle and/or its affiliates. All rights reserved.

Oracle is a registered trademark of Oracle Corporation and/or its
affiliates. Other names may be trademarks of their respective
owners.

Type 'help;' or '\h' for help. Type '\c' to clear the current input statement.

mysql> show databases;
+-----+
| Database |
+-----+
| atm_machine |
| bank_management_system |
| class |
| information_schema |
| library_app |
| moorthy |
| motel |
| mukesh |
| mysql |
| pay |
| performance_schema |
| school_db |
| son |
| vicki |
+-----+
14 rows in set (0.05 sec)

mysql> use school_db;
Database changed
mysql> show tables;
+-----+
| Tables_in_school_db |
+-----+
| marksheet |
| students |
+-----+
2 rows in set (0.03 sec)
```



```
mysql> desc students;
```

Field	Type	Null	Key	Default	Extra
Roll_Number	int	NO	PRI	NULL	auto_increment
name	varchar(100)	YES		NULL	
attendance	int	YES		NULL	
hours_studied	float	YES		NULL	
weekly_study_hours	float	YES		NULL	
previous_score	float	YES		NULL	
assignments_completed	int	YES		NULL	
stress_level	int	YES		NULL	
learning_style	varchar(20)	YES		NULL	
extracurriculars_involved	int	YES		NULL	
goal_score	float	YES		NULL	
score	float	YES		NULL	

```
12 rows in set (0.00 sec)
```

```
mysql> desc marksheet;
```

Field	Type	Null	Key	Default	Extra
Roll_Number	int	NO	PRI	NULL	auto_increment
Name	varchar(100)	YES		NULL	
Stream	varchar(50)	YES		NULL	
Physics	float	YES		NULL	
Maths	float	YES		NULL	
Chemistry	float	YES		NULL	
English	float	YES		NULL	
Computer_Science	float	YES		NULL	
Artificial_Intelligence	float	YES		NULL	
Biology	float	YES		NULL	
Accountancy	float	YES		NULL	
Economics	float	YES		NULL	
Business_Studies	float	YES		NULL	
IP	float	YES		NULL	
Total_Marks	float	YES		NULL	
Percentage	float	YES		NULL	
Result	varchar(10)	YES		NULL	

```
17 rows in set (0.00 sec)
```



```
mysql> select * from students;
```

```
+-----+-----+-----+-----+
| Roll_Number | name      | attendance | hours_studied |
| extracurriculars_involved | goal_score | score |
+-----+-----+-----+-----+
|          1 | MERCURY  |          80 |          7.8 |
|          2 | GANDHI   |          90 |          9.8 |
|          3 | RADIUM   |          75 |          8.6 |
|          4 | SANJAY   |          50 |          10.8 |
|          5 |          |          30 |          500 |
+-----+-----+-----+-----+
4 rows in set (0.00 sec)
```

```
+-----+-----+-----+-----+
weekly_study_hours | previous_score | assignments_completed | stress_level | learning_style
+-----+-----+-----+-----+
|          7.5 |          413 |          28 |          7 | Kinesthetic
|          8.7 |          470.6 |          30 |          3 | Visual
|          6.6 |          413 |          25 |          7 | Visual
|          9.9 |          498.9 |          30 |          10 | Kinesthetic
+-----+-----+-----+-----+
```

```
mysql> select * from marksheet;
```

Roll_Number	Name	Stream	Physics	Maths	Chemistry	English	Computer_Science
IP	Total_Marks	Percentage	Result				
1	MERCURY	Science	75	80	85	90	95
90	944	85.8182	Pass				
2	GANDHI	Science	75	80	85	90	95
90	944	85.8182	Pass				
3	RADIUM	Science	75	80	85	90	95
90	944	85.8182	Pass				
4	SANJAY	Science	75	80	85	90	95
90	944	85.8182	Pass				
Artificial_Intelligence Biology Accountancy Economics Business_Studies							
		88	92	87	84		78
		88	92	87	84		78
		88	92	87	84		78
		88	92	87	84		78

3.5 EXPLANATION ABOUT THE (SRP) PROGRAMME

This program is designed to create, manage, and analyze student performance data stored in a MySQL database. It provides functionalities to add student information, display and search for specific student records, train a predictive model for student scores, and generate a marksheets. The program uses Python libraries like **pandas**, **MySQL connector**, **SQLAlchemy**, **scikit-learn**, and **tabulate** to handle data processing, machine learning, and database interactions.

Libraries Used and Their Purpose

1. **pandas**: Essential for data manipulation, loading data from SQL tables, and formatting data for machine learning models. It's also used here to structure and display the data neatly in tables.
2. **mysql.connector**: A library for connecting to and communicating with MySQL databases. It allows the program to perform SQL operations like creating tables, inserting data, and querying records.
3. **SQLAlchemy**: An ORM library that abstracts SQL operations to a Pythonic approach. Here, it manages database connections and lets us perform SQL operations easily using Python objects.
4. **scikit-learn**:
 - **train_test_split**: Used to split the data into training and testing sets, allowing the model to be tested for accuracy.
 - **LinearRegression**: A simple yet powerful algorithm from scikit-learn's `linear_model` module used here to predict student scores based on their historical performance data.
 - **mean_squared_error and r2_score**: Evaluation metrics that measure the accuracy of the model's predictions.
5. **tabulate**: Converts data into a formatted string for a more readable display in the console, used here for showing database records in table format.
6. **warnings**: Specifically used here to ignore warnings related to the evaluation metrics in cases where not enough data is available to calculate these scores.

Functions Explained in Detail

1. Database Initialization Functions

create_database()

- This function ensures that the database (`school_db`) exists before proceeding.
- If it doesn't exist, it creates it by connecting to MySQL without specifying a database and then executing a SQL command to create `school_db`.

create_student_table()

- Ensures that the **students** table is created in `school_db`.
- Defines the table with columns for **student data**: `Roll_Number` (auto-incremented), `name`, `attendance`, `hours_studied`, and other metrics related to the student's study habits and goals.

`create_marksheet_table()`

- Similar to `create_student_table`, but for the **MARKSHEET** table, which holds data related to a student's **subject-wise scores, total marks, percentage, and result status**.
- This table will be populated with marksheet data using the `create_marksheet()` function later in the program.

2. Database Connection and Helper Functions

`create_db_connection()`

- Creates and returns a connection object to interact with `school_db` via SQLAlchemy.
- This function allows other functions to easily connect to the database without repeating connection code.

`display_table()`

- Retrieves all data from the **students** table and displays it in a tabulated format using `tabulate` if any data exists.
- Provides a quick way to view the entire **students** table directly in the console.

`search_student_by_roll(Roll_Number)`

- Searches for a student by their unique roll number in the **students** table.
- Uses the roll number to retrieve and display the student's data in a neat, tabulated format. If no record is found, it outputs an appropriate message.

3. Data Entry Functions

`insert_student_data()`

- Inserts a new student record into the **students** table.
- Accepts various parameters related to the student's performance and personal metrics, and inserts this data into the database using SQLAlchemy's `execute` method.
- Provides error handling for cases where the data might fail to insert.

`get_student_data()`

- Prompts the user to enter a range of metrics about a student, such as name, attendance, weekly study hours, stress level, and more.
- Calls `insert_student_data()` to insert the inputted data into the **students** table.
- This function is crucial for adding new data, which will be used for training the predictive model.

4. Predictive Model Training and Prediction Functions

`fetch_student_data()`

- Retrieves specific columns from the **students** table needed for training the model. Returns a pandas DataFrame containing the data, which will be split into features (x) and target (y) for model training.

train_and_predict()

- Trains a **linear regression model** using student data from the database to predict a student's score.
- Process:
 1. **Fetch data** from `fetch_student_data()`.
 2. **Encode learning styles** to numerical values for the model.
 3. **Split data** into features (x) and target (y), where y is the **score**.
 4. **Train-test split**: Divides the data into training and testing sets.
 5. **Train model**: Fits a `LinearRegression` model on the training data.
 6. **Predict**: Uses the model to predict scores on the testing set.
 7. **Evaluation**: Computes and displays Mean Squared Error and R-squared Score to assess model accuracy.
- **User Input Prediction**:
 - After training, prompts for new student data and predicts their score.
 - Calls `early_warning_system()` if the predicted score falls below a threshold.

early_warning_system(predicted_score, threshold=50)

- Checks if the predicted score is below a specified threshold (default is 50).
- If the score is low, prints a warning message and a notification for parents and teachers.

5. Marksheet Generation

create_marksheet()

- Generates a **marksheet** for each student in the **students** table.
- Steps:
 1. Fetches each student's data from `students`.
 2. **Default Score Assignment**: For simplicity, it assigns default scores for each subject (e.g., Physics: 75, Maths: 80, etc.).
 3. **Total and Percentage Calculation**: Sums the scores and calculates the percentage for each student.
 4. **Result Determination**: Sets `Result` as "Pass" if the percentage is above a passing threshold (e.g., 33%), otherwise "Fail".
 5. Inserts each student's marksheet data into the **MARKSHEET** table.

6. Main Program Menu

- **Option 1: Add Student Data** – Calls `get_student_data()` to prompt the user for new student details and store them in the **students** table.
- **Option 2: Display Students Table** – Calls `display_table()` to show all current student records in the **students** table.
- **Option 3: Search Student by Roll Number** – Prompts for a roll number and calls `search_student_by_roll()` to look up and display that student's information.
- **Option 4: Train Model and Predict Score** – Calls `train_and_predict()` to train the predictive model and predict scores for new student data entered by the user.
- **Option 5: Create Marksheet** – Calls `create_marksheet()` to generate marksheets for all students based on preset subject scores and calculates the total, percentage, and result.

1. Database and Table Creation

- **Purpose:** Ensures the required database and tables are set up to store student data and results.
- **Execution:**
 - **Database Creation (`create_database`):** When the program starts, it first verifies if the database (`school_db`) exists. If not, it creates `school_db`.
 - **Table Creation:**
 - **`create_student_table()`**: The program checks if the `students` table exists. If it doesn't, it creates the table with columns for student data attributes such as attendance, hours studied, and stress level.
 - **`create_marksheet_table()`**: Similarly, the program verifies or creates the `MARKSHEET` table, designed to store subject-wise scores, total marks, percentage, and pass/fail results.
- **Outcome:** Both tables, `students` and `MARKSHEET`, are ready for data storage and retrieval.

2. Adding New Student Data

- **Purpose:** Collects new student information and stores it in the `students` table, which provides training data for the predictive model.
- **Execution:**
 - **User Input Collection (`get_student_data()`):**
 - The program prompts the user for detailed information about a student, including fields such as name, attendance percentage, hours studied, stress level, and goal score.
 - **Data Insertion (`insert_student_data()`):**
 - After the user inputs data, the program calls `insert_student_data()` to insert this data into the `students` table.
 - SQLAlchemy is used to connect to the database and execute the `INSERT` command to store each student record.
 - **Error Handling:**
 - If any issue arises during data insertion, such as invalid data types or database connection issues, an error message is displayed.
 - **Outcome:** The `students` table now holds the entered student data, ready for further use.

3. Viewing and Searching Student Data

- **Purpose:** Allows users to view all students' records or look up specific student details using their roll number.
- **Execution:**
 - **Display All Records (`display_table()`):**
 - When this option is selected, `display_table()` retrieves all data from the `students` table and presents it in a tabulated format using `tabulate`.
 - **Search by Roll Number (`search_student_by_roll()`):**
 - The program prompts the user to enter a roll number, which is then used to search the `students` table for a specific record.
 - If a matching record is found, it is displayed in a table format; otherwise, a "No student found" message appears.
- **Outcome:** Users can view student data as needed, aiding in data verification and understanding student records.

4. Training the Predictive Model

- **Purpose:** Builds a linear regression model to predict student scores based on existing performance data.
- **Execution:**
 - **Data Retrieval (`fetch_student_data()`):**
 - The program fetches specific columns from the `students` table needed for prediction (e.g., attendance, hours studied, stress level).
 - **Data Preparation and Encoding:**
 - Converts the `learning_style` column into numerical values (e.g., Visual = 1, Auditory = 2, Kinesthetic = 3) to make it suitable for modeling.
 - **Splitting Data:**
 - Divides the dataset into features (X) and target (y), where y is the `score`.
 - Uses `train_test_split` to split data into training and testing sets, ensuring 80% of data is used for training and 20% for testing.
 - **Model Training:**
 - A `LinearRegression` model is instantiated and trained on the training dataset (`X_train`, `y_train`).
 - **Model Evaluation:**
 - After training, the model's predictions (`y_pred`) are evaluated on the test dataset (`X_test`) using Mean Squared Error (MSE) and R-squared score (R^2) to measure accuracy.
 - MSE quantifies the average squared difference between predicted and actual scores, while R^2 indicates the proportion of variance explained by the model.
- **Outcome:** A trained model that can be used to predict scores based on student data. Evaluation metrics (MSE, R^2) give insight into model accuracy.

5. Predicting Scores for New Students

- **Purpose:** Predicts a new student's score using the trained model based on their study habits, learning style, and performance data.
- **Execution:**
 - **Data Input:**
 - The user inputs details about the new student, similar to the data collected in Step 2.
 - This new data is formatted to match the structure of the training data, ensuring compatibility with the trained model.
 - **Prediction:**
 - The model predicts the `score` based on the input data.
 - **Early Warning System (`early_warning_system()`):**
 - The program evaluates the predicted score to determine if it falls below a threshold (e.g., 50).
 - If the predicted score is low, a warning message is generated, suggesting that the student may be at risk of underperforming.
 - This feature is a basic intervention measure, notifying parents or teachers to assist the student.
- **Outcome:** Provides a score prediction for the new student, with a warning if the predicted score is below the specified threshold.

6. Generating Marksheet for Students

- **Purpose:** Creates a detailed marksheet for each student, complete with subject-wise scores, total marks, percentage, and pass/fail status.
- **Execution:**
 - **Data Fetching:**
 - The program retrieves each student's details from the `students` table.
 - **Score Assignment:**
 - Assigns preset scores for each subject (e.g., Physics: 75, Maths: 80), simulating an actual marksheet.
 - These scores are only examples and could be modified based on the student's record or the program's requirements.
 - **Calculations:**
 - **Total Marks:** Summed across all subjects.
 - **Percentage:** Calculated as the total score divided by the number of subjects.
 - **Pass/Fail Status:** Set as "Pass" if the percentage is above a threshold (e.g., 33%) and "Fail" if it is below.
 - **Insertion into MARKSHEET Table:**
 - Inserts the compiled marksheet information for each student into the `MARKSHEET` table, which holds the full record of scores, percentages, and results.
- **Outcome:** A comprehensive marksheet for each student is stored in the `MARKSHEET` table, summarizing their performance across subjects and indicating their pass/fail status.

7. Program Options (Main Menu)

- **Purpose:** Provides a menu-driven interface, allowing users to perform each of the above actions based on their needs.
- **Options Available:**
 - **Add Student Data:** Calls `get_student_data()` to input and store new student information in the `students` table.
 - **Display Students Table:** Calls `display_table()` to view all records in the `students` table.
 - **Search Student by Roll Number:** Calls `search_student_by_roll()` to locate a student by their roll number.
 - **Train Model and Predict Score:** Initiates `train_and_predict()` to train the model, evaluate its performance, and predict a new student's score.
 - **Create Marksheet:** Calls `create_marksheet()` to generate and store marksheets for all students.
 - **Exit:** Terminates the program.
- **Outcome:** A user-friendly way to access all functionalities, making it easier to manage and analyze student data.




3.6 BASIC NEEDS

Python Interpreter:

What it is: Python is an interpreted language, meaning it runs on an interpreter that converts your code into a format the computer understands.

What to do:

SCAN and Download the latest version of Python from the official website 


Install Python by following the installation instructions for your operating system (Windows, macOS, or Linux).

Important: During installation, ensure you check the option "Add Python to PATH" (on Windows) so you can run Python from the command line.

Install MySQL Server

MySQL is a popular open-source relational database management system (RDBMS). You'll need to install the MySQL Server to store, retrieve, and manage your databases.

Download MySQL:

SCAN and Download the official MySQL download page: 

Download the MySQL Installer for your operating system (Windows, macOS, or Linux).



PYTHON DOWNLOAD



MYSQL DOWNLOAD

Install MySQL:

During the installation, choose the "Developer Default" option to install the MySQL server, MySQL Workbench, and other tools.

Set up a root password for the MySQL server during installation (remember this password for later)..

Testing the Connection between Python and MySQL:

```
1 import mysql.connector
2
3 # Establish a connection to the MySQL database
4 mydb = mysql.connector.connect(
5     host="localhost",      # Use 'localhost' if MySQL is running on your local machine
6     user="root",           # The username you set during installation (usually 'root')
7     password="your_password", # The password you set during installation
8     database="test_db"     # Name of the database you want to connect to (or create one)
9 )
10
11 # Create a cursor object
12 mycursor = mydb.cursor()
13
14 # Execute a simple query
15 mycursor.execute("SHOW DATABASES")
16
17 # Print the result
18 for db in mycursor:
19     print(db)
20
```

GENERAL POINTS OF STUDENTS RESULTS PREDICTOR

BENEFITS OF USING STUDENTS RESULTS PREDICTOR :

Using a student results predictor that incorporates Artificial Intelligence (AI) offers a range of benefits that enhance the educational experience for students, teachers, and administrators. Here are the key advantages:

1. Improved Accuracy in Predictions

- **Benefit:** AI algorithms can analyze vast amounts of data and identify complex patterns that may not be apparent through traditional statistical methods. This results in more accurate predictions regarding student performance and potential outcomes.
- **Impact:** More reliable predictions help educators target interventions effectively and improve overall academic performance.

2. Personalized Learning Experiences

- **Benefit:** AI can tailor learning pathways based on individual student data, such as learning styles, study habits, and previous performance. It allows for adaptive learning systems that adjust content delivery to match student needs.
- **Impact:** Students receive personalized support, which can lead to enhanced engagement and better learning outcomes.

3. Real-Time Data Analysis

- **Benefit:** AI systems can analyze data in real time, providing immediate feedback to students and educators. This allows for timely interventions and adjustments to teaching strategies based on current performance metrics.

- **Impact:** Prompt feedback enhances learning agility and allows students to address issues as they arise.

4. Enhanced Decision-Making Capabilities

- **Benefit:** AI tools provide insights and recommendations based on data analysis, helping educators make informed decisions about curriculum design, resource allocation, and intervention strategies.
- **Impact:** Data-driven decision-making leads to more effective teaching methods and better student support programs.

5. Scalability Across Educational Institutions

- **Benefit:** AI systems can be easily scaled and adapted to various educational settings, from individual classrooms to entire school districts. They can process large volumes of student data efficiently.
- **Impact:** This scalability ensures that all students, regardless of the institution's size, benefit from advanced predictive analytics.

6. Early Warning Systems for At-Risk Students

- **Benefit:** AI can identify students who are at risk of underperforming by analyzing multiple indicators (e.g., attendance, engagement, and stress levels) to trigger alerts for educators and administrators.
- **Impact:** This proactive approach allows for early interventions, improving retention rates and student success.

7. Enhanced Engagement Through Interactive Tools

- **Benefit:** AI-powered platforms can include interactive elements, such as chatbots and personalized learning apps, that engage students in their learning process.
- **Impact:** Increased engagement leads to higher motivation and participation in academic activities.

8. Efficient Administrative Processes

- **Benefit:** AI can automate administrative tasks, such as grading and tracking attendance, allowing educators to focus more on teaching and student interaction.
- **Impact:** This increases efficiency within educational institutions and allows for more time spent on student-centric activities.

9. Identification of Effective Teaching Strategies

- **Benefit:** AI can analyze which teaching methods are most effective based on student performance data, helping educators refine their instructional strategies.
- **Impact:** Continuous improvement in teaching practices leads to enhanced student learning experiences and outcomes.

10. Support for Diverse Learning Needs

- **Benefit:** AI can identify diverse learning needs among students, including those requiring special education services, and help design customized interventions.
- **Impact:** This inclusivity ensures that all students receive the necessary support to succeed academically.

11. Longitudinal Performance Tracking

- **Benefit:** AI systems can track student performance over time, providing insights into academic progress and trends. This longitudinal data helps in evaluating the long-term effectiveness of educational programs.
- **Impact:** Educators can make informed adjustments to programs and curricula based on comprehensive performance data.

12. Greater Parental Involvement

- **Benefit:** AI-powered tools can provide parents with insights into their child's performance, allowing them to engage more effectively in their education.
- **Impact:** Increased communication and collaboration between schools and families enhance student support and motivation.

13. Research and Development Insights

- **Benefit:** The data collected by AI systems can contribute to educational research, helping institutions understand factors influencing student success and developing best practices.
- **Impact:** Research-driven improvements can lead to systemic changes that benefit the entire educational community.

14. Cost-Effectiveness

- **Benefit:** AI can reduce costs related to tutoring and other support services by predicting needs accurately and enabling targeted interventions.
- **Impact:** Schools can allocate resources more effectively, ultimately enhancing the quality of education without significant increases in expenditure.

In summary, leveraging AI in a student results predictor not only enhances the accuracy and effectiveness of academic interventions but also creates a more personalized and supportive educational environment. The integration of AI facilitates informed decision-making, efficient resource allocation, and a focus on continuous improvement in teaching and learning practices.

CONCLUSION

Conclusion of the Student Results Predictor Project Utilizing AI

The Student Results Predictor project represents a significant advancement in the realm of educational technology, merging the power of Artificial Intelligence (AI) with the imperative of student academic success. This project has been meticulously designed to not only forecast student performance but also to enhance the overall educational experience by providing data-driven insights that benefit students, educators, and administrators alike.

One of the core objectives of the project is to employ predictive analytics to identify patterns and trends in student data. By analyzing various factors such as attendance, hours studied, previous scores, and stress levels, the AI model provides accurate predictions of future academic performance. This capability allows educators to intervene proactively, offering support to students at risk of underperforming before issues escalate. The implementation of such an early warning system can be transformative; it fosters a supportive environment that prioritizes student well-being and academic achievement.

Moreover, the project emphasizes the importance of personalized learning experiences. AI's ability to tailor educational pathways according to individual student needs is a significant benefit. By considering diverse factors—such as learning styles and extracurricular involvement—the predictor aids in crafting a unique learning journey for each student. This personalization not only enhances engagement but also motivates students, resulting in improved academic outcomes and a deeper investment in their education.

The integration of AI within the student results predictor also streamlines administrative processes. By automating tasks such as data analysis and reporting, educators can dedicate more time to teaching and mentoring. This shift not only improves the efficiency of educational institutions but also enriches the quality of student-teacher interactions, ultimately fostering a more enriching academic environment.

Additionally, the project underscores the importance of data-driven decision-making in education. The insights derived from AI analytics empower educators and administrators to make informed choices regarding curriculum design, resource allocation, and intervention strategies. By understanding the effectiveness of various teaching methods, educational institutions can continuously refine their approaches, leading to enhanced teaching practices and improved student outcomes.

Furthermore, the project fosters greater parental involvement by providing insights into student performance. This engagement helps create a collaborative educational atmosphere where parents, educators, and students work together towards common academic goals. Enhanced communication channels ensure that parents are informed and can support their children effectively, which is crucial for student motivation and success.

In conclusion, the Student Results Predictor project exemplifies the transformative potential of AI in education. By harnessing the capabilities of AI for predictive analytics, the project not only forecasts student performance but also cultivates a supportive, personalized, and efficient educational environment. The benefits of early intervention, tailored learning experiences, data-driven decision-making, and enhanced stakeholder engagement highlight the profound impact of this project.

BIBLIOGRAPHY

- <https://www.thinkitive.com/blog/10-benefits-of-ai-in-technology-and-business/>
- https://www.freepik.com/search?format=search&img=1&last_filter=img&last_value=1&query=ARTIFICIAL+INTELLIGENCE+WITH+STUDENTS
- <https://chatgpt.com/c/67208f45-c150-800c-9b47-2e4b1a3c6b63>
- <https://www.ibm.com/topics/artificial-intelligence>
- <https://slejournal.springeropen.com/articles/10.1186/s40561-022-00192-z>
- <https://www.analyticsvidhya.com/blog/2023/04/student-performance-analysis-and-prediction/>
- <https://github.com/mohammedAljadd/students-performance-prediction>