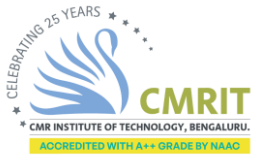
HACKATHON PROJECT REPORT

Project Title

"Student Performance Predictor"



Team Members

Ashirwad (1CR23CD011)

C Vishwak Sena (1CR23CD017)

Aditya Mule (1CR23CD036)

Submitted For

(Full Stack Odyssey)

Department of Computer Science Engineering (Data Science)

CMR Institute of Technology, Bengaluru

Abstract

Educational institutions traditionally depend on exam-based evaluation, which reveals performance only after assessments. This limits opportunities for early assistance to struggling learners. The **Student Performance Predictor** bridges this gap by using machine learning to forecast academic outcomes much earlier.

The predictor considers multiple academic and behavioral factors including attendance, internal marks, study hours, consistency, assignments, and extracurricular activity. These parameters collectively help estimate whether a student is likely to perform well or may need academic support.

The system, built during a hackathon, uses a Random Forest classifier and includes a responsive web interface for quick data entry and instant feedback. It aims to support institutions in monitoring academic progress, identifying at-risk students early, and promoting data-driven academic planning. Overall, the project showcases an efficient, deployable ML solution for education-focused analytics.

Contents

Chapter 1 Introduction

1.1 Background

Educational Data Mining (EDM) helps institutions analyze learning behavior, performance indicators, and patterns that influence student success. With increasing data availability, analytics-driven decision-making is becoming essential in academic environments.

1.2 Problem Statement

Performance issues are usually identified only after internal tests or final exams. This delay reduces the effectiveness of academic intervention, leaving many students without the required support at the right time.

1.3 Objectives

- Predict student performance using machine learning.
- Provide a simple and intuitive web-based interface.
- Enable early academic intervention and personalized support.
- Demonstrate a scalable ML workflow suitable for real-world use.

1.4 Scope

The system can be integrated into schools, colleges, or coaching centers. It can also be enhanced with dashboards, automated reports, and integration with ERP systems.Allows users to early detection of passing grade and helping to improve future performance

Chapter 2

System Overview

2.1 Key Components

- **Dataset Module** – Stores and preprocesses student data.
- **Feature Engineering** – Extracts meaningful performance predictors.
- **ML Model** – Multiple Machine-Learning model used for predicting grade ,confidence level and probabilty of passing.
- **Web UI** – User-friendly frontend for students and faculty to monitor their performance.

2.2 DataBase Architecture

The user enters student details through the interface. The backend processes the data, passes it to the model, and returns the predicted outcome along with supportive insights.

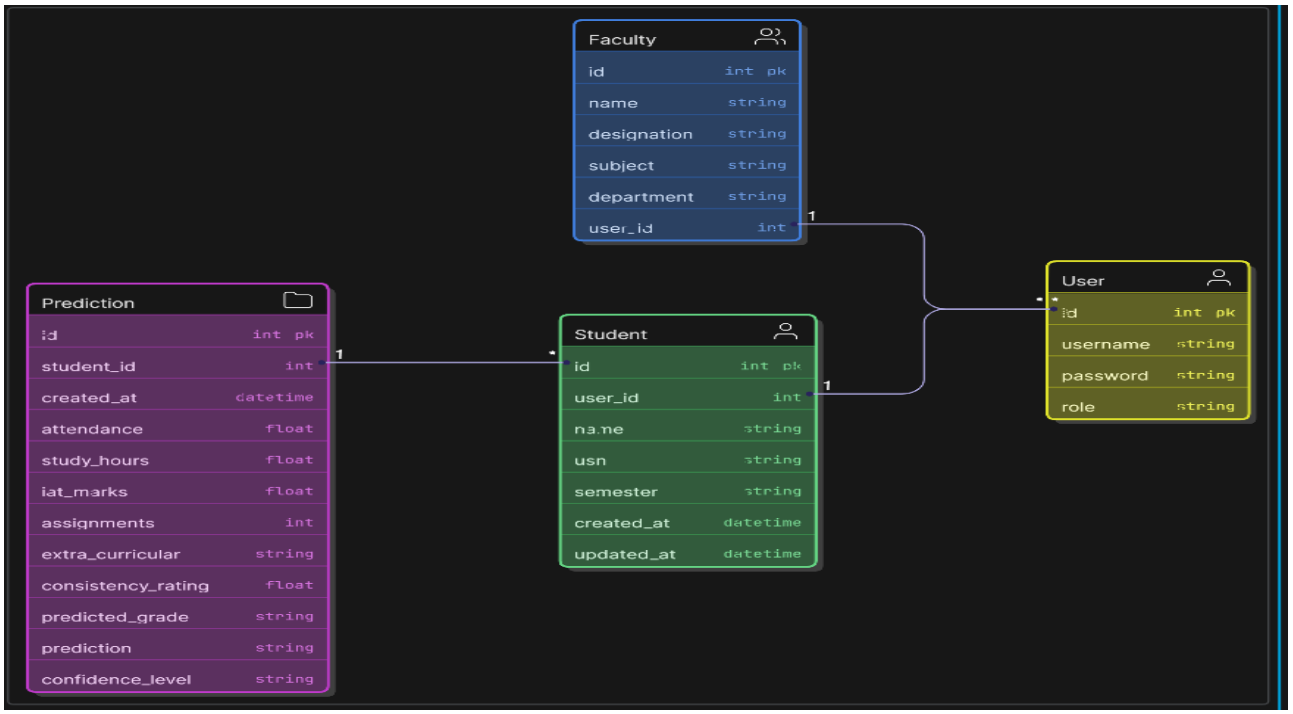


Figure 2.1: System Architecture

Chapter 3

Implementation

3.1 Technologies Used

The project uses Python for model training and backend deployment.For frontend it is utilising React.js and Tailwind for User interactive pages. Supporting libraries include Pandas for data processing and Scikit-learn for model implementation.Flask for backend and machine learning model deployment.And sqlite is used for database which makes the predictions availabe for future reference.

3.2 Repository Architecture

- **Backend/** – Handles prediction requests and data processing and also allows to serve api calls to download prediction report and requests such as login,sugnup,etc.
- **ML/** – Contains training scripts and saved models.Multiple models are trained to predict the grade ,confindence level and passing probability.
- **frontend/** – Simple and attractive UI to easily navigate through the platform.Easy form based pages to predict the passing grade and probabilty.

GitHub language statistics:

- JavaScript: 74.7%
- Python: 24.2%
- Others: 1%

3.3 Dataset Processing

The dataset undergoes cleaning, handling missing values, feature scaling, and label encoding. It is then split into training and testing sets to evaluate model accuracy.Here we have used 80% of data to train the model and 20% to test the model in all the 3 models.

3.4 Training Code Example

```
print("🔧 Training ML models in backend environment…")

# ------------------------------------------------------
# LOAD YOUR DATASET
# ------------------------------------------------------
df = pd.read_csv("dataset.csv")  # <-- change this to your actual
file name

# Auto-detect target columns (same logic as notebook)
cols_lower = {c.lower(): c for c in df.columns}

def find_col(text):
    for c in df.columns:
        if all(t in c.lower() for t in text):
            return c
    return None

grade_col = find_col(["predicted", "grade"])
conf_col = find_col(["confidence"])
predictor_col = find_col(["predictor"]) or find_col(["pass"])

target_cols = [grade_col, conf_col, predictor_col]

print("Targets detected:", target_cols)

# ------------------------------------------------------
# FEATURES / PREPROCESSOR
# ------------------------------------------------------
X = df.drop(columns=target_cols)

num_cols = X.select_dtypes(include=[np.number]).columns.tolist()
cat_cols = X.select_dtypes(exclude=[np.number]).columns.tolist()

preprocessor = ColumnTransformer(
    transformers=[
        ("num", StandardScaler(), num_cols),
        ("cat", OneHotEncoder(handle_unknown="ignore"),
cat_cols),
    ]
)

# ------------------------------------------------------
# TRAIN FUNCTION
# ------------------------------------------------------
def train_best_model(target):
    y = df[target]

    # Determine classification vs regression
    is_numeric = np.issubdtype(y.dtype, np.number)
    problem = "regression" if (is_numeric and y.nunique() > 10)
else "classification"

    print(f"\n🎯 Training: {target}  ({problem})")

    if problem == "classification":
```

Chapter 4

Frontend UI

4.1 Interface

The user interface allows quick input of essential academic parameters. It is optimized for speed, responsiveness, and simplicity—ideal for hackathon demonstrations.

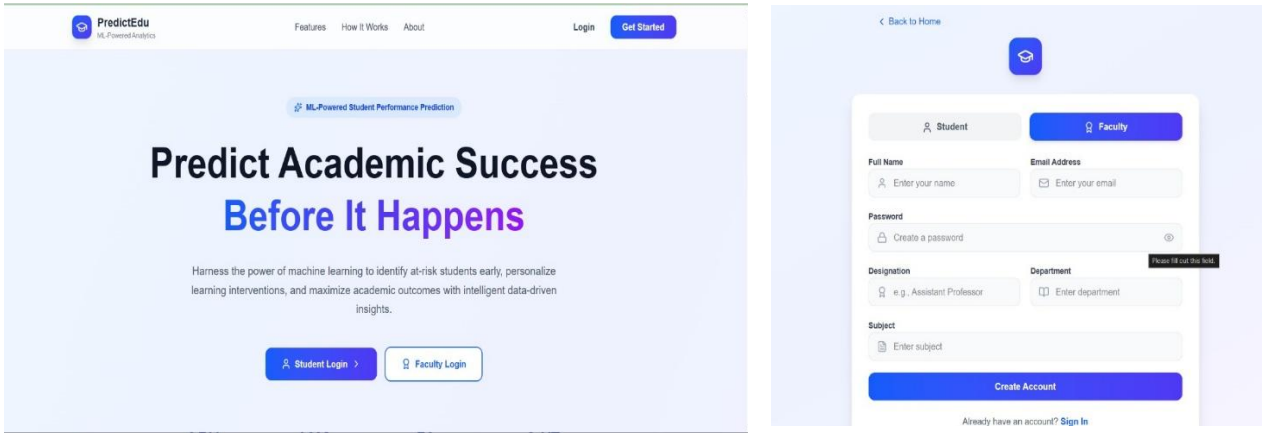


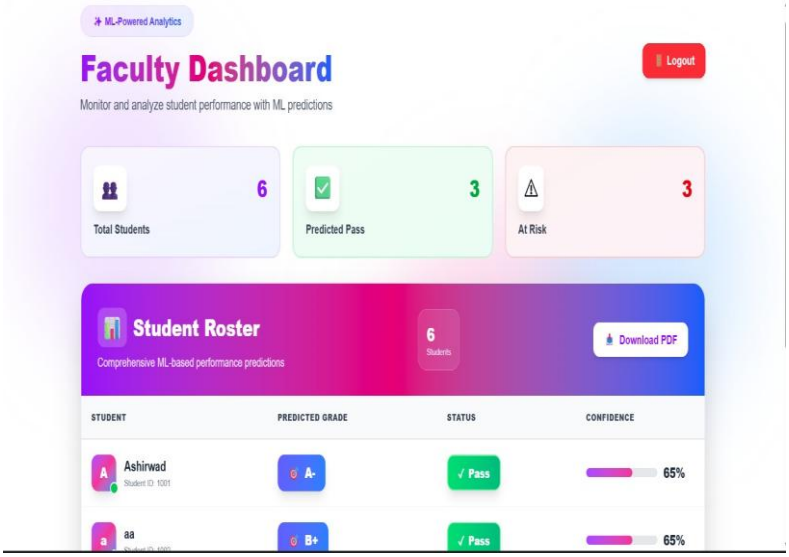
Figure 4.1: User Interface

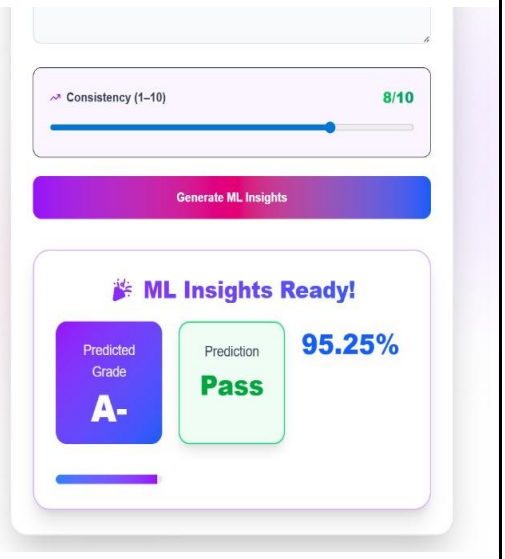
4.2 User Flow

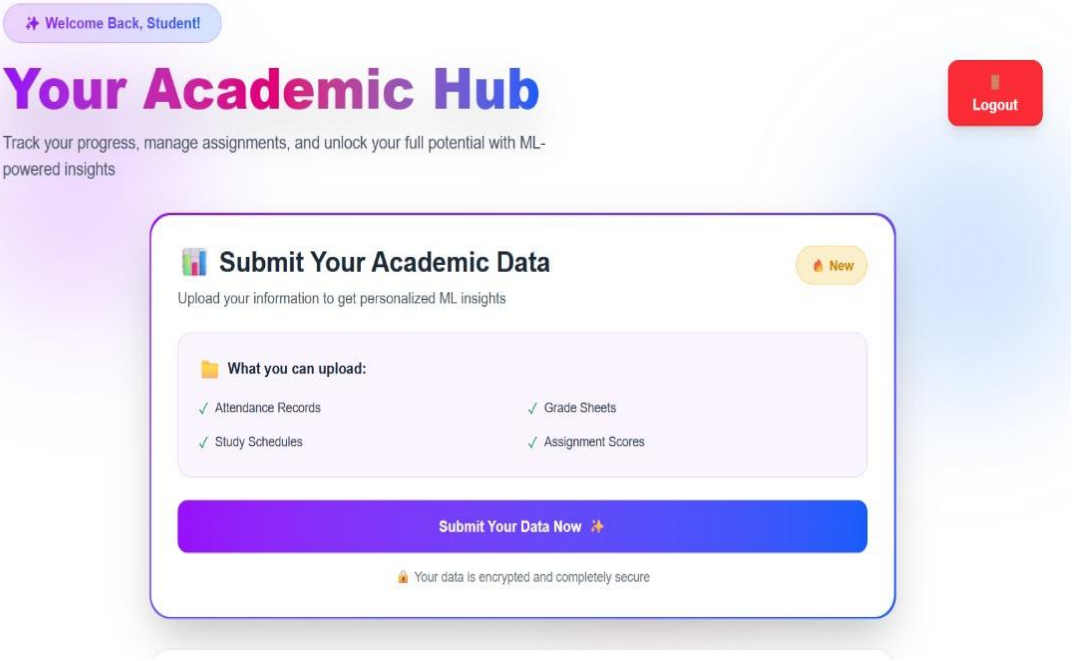
- User land to the home page and register itself
- After registration ,user needs to login.
- For students
    - User enters student details.
    - Data is sent to the backend API.
    - Model predicts performance.
    - Results are displayed instantly
- For Faculty

- It need to login.
- Next user can see all the student data and prediction.
- The user can also download it as a pdf file.

4.3 UI







Chapter 5

Result and Conclusion

5.1 Result

The project demonstrates how ML can enhance academic monitoring systems by providing early, actionable insights. Its lightweight architecture makes it suitable for quick deployment and real-world use. The model achieved an accuracy of **92-97%** depending on dataset split variations, demonstrating robust performance under limited data.

5.2 Conclusion

- Attendance has a major impact on predicted performance.
- Past academic performance strongly influences outcomes.
- Consistency and study hours contribute to prediction strength.

References

1. Scikit-learn — `https://scikit-learn.org/`
2. Pandas — `https://pandas.pydata.org/`
3. GitHub Repository — `https://github.com/ASHIR-WAD/Student-Performance-Predictor`