**MLG382  Guided project**

**Group H**

Bianca Grobler 600537  
Adolph Jacobus van Coller 601005  
Caydan Frank 578131  
Renaldo Jardim 601333

Predictive Analysis of Student Academic Performance Using Machine Learning and Deep Learning Models

# Table of Contents

[Table of Contents 1](#_Toc2141193733)

[Executive Summary 2](#_Toc1693201370)

[Introduction 3](#_Toc902276474)

[Problem Statement 4](#_Toc1365807231)

[Hypothesis Generation 5](#_Toc185111138)

[System Setup and Data Loading 6](#_Toc465189055)

[Data Understanding 7](#_Toc941233094)

[Exploratory Data Analysis (EDA) 8](#_Toc185854128)

[Data Preprocessing 11](#_Toc2044009150)

[Feature Engineering 12](#_Toc1529492924)

[Model Building – Machine Learning 13](#_Toc1436020457)

[Model Building – Deep Learning 16](#_Toc1492958312)

[Deployment 18](#_Toc2144936996)

[Conclusion 21](#_Toc300448892)

# Executive Summary

BrightPath Academy is an innovative high school that values both academic achievement and holistic development. Despite its commitment to supporting every learner, the school faces several challenges, including delayed identification of at-risk students, limited tools for tailoring interventions, and an unclear understanding of the influence of extracurricular activities on student academic outcomes.

This project leverages the Student\_performance\_data dataset provided to build predictive models that classify students' academic performance (GradeClass). By employing both machine learning and deep learning algorithms, the project aims to provide timely, data-driven insights that can help educators develop more personalized and effective support strategies. Ultimately, the goal is to facilitate early intervention, enhance academic outcomes, and support holistic student growth through the intelligent use of data.

# Links

## GitHub Repository

<https://github.com/CL-Frank/MLG382_GuidedProject>

## Render Application

<https://mlg382-guidedproject-1.onrender.com>

# Introduction

## About BrightPath Academy

BrightPath Academy is a forward-thinking secondary school located in a diverse urban environment. The school emphasizes academic excellence, personal development, and active participation in extracurricular activities. It seeks to equip students not only with strong academic foundations but also with the interpersonal and social skills necessary for lifelong success.

## Project Overview

To align with its mission, BrightPath Academy seeks to transform its vast repository of student data into actionable insights. This project is cantered on developing a predictive analytics system capable of identifying academic performance patterns, predicting student outcomes, and informing timely interventions. The system will utilize various student attributes like; demographics, study habits, parental support, and extracurricular involvement, to predict each student's GradeClass.

## Objectives

The primary goal of this project is to develop a data-driven system that enables BrightPath Academy to enhance its academic support strategies through predictive analytics. The specific objectives are as follows:

* Develop a Predictive Classification Model.  
  Build machine learning and deep learning models to accurately classify students into predefined academic performance categories (GradeClass) based on their GPA. This classification will help in segmenting students into actionable risk levels for academic performance.
* Enable Early Identification of At-Risk Students.  
  Use predictive models to flag students who are at risk of achieving low grades early in the academic year. This early warning system will allow educators and support staff to intervene before students fall significantly behind.
* Analyse Key Performance Indicators.  
  Determine which variables (for example, study habits, parental involvement, extracurricular activities, absences) most significantly influence academic success. This analysis will provide deeper insights into the factors driving student outcomes at BrightPath Academy.
* Evaluate the Effectiveness of Extracurricular Activities.  
  Explore and quantify the relationship between student involvement in activities like music, sports, and volunteering and their academic performance. These insights will help assess the educational value of non-academic engagements.
* Facilitate Data-Driven Decision Making for Educators.  
  Present model outcomes and analytics through a clear and interactive dashboard. This will empower teachers, academic advisors, and school counsellors to make informed, real-time decisions tailored to each student's needs.
* Ensure Interpretability and Transparency of Models.  
  Select and evaluate models not only for their performance but also for their interpretability, ensuring stakeholders understand why specific predictions are made. This is critical in educational settings where decisions must be justified and explainable.
* Deploy a Scalable Solution.  
  Host the final predictive system on a cloud-based platform (Render) using Dash, ensuring accessibility, scalability, and ease of use for school staff without requiring specialized technical knowledge.
* Promote Continuous Improvement Through Data Feedback Loops.  
  Lay the groundwork for a system that evolves with new data inputs over time, enabling BrightPath Academy to continuously refine its intervention strategies and adapt to changing student needs.

# Problem Statement

BrightPath Academy is dedicated to fostering academic excellence while supporting the holistic development of its students. However, the school faces several data and insight-related challenges that limit its ability to deliver timely and targeted academic interventions. Despite having access to a rich set of student data, educators often struggle to convert this information into actionable strategies that address individual student needs.

One of the most pressing issues is the delayed identification of at-risk students. Academic struggles are only recognized in the latter part of the school year, reducing the effectiveness of intervention efforts. This delay can result in missed opportunities to provide timely support that could positively influence student outcomes.

Additionally, the school lacks a systematic approach for personalizing support strategies such as tutoring or mentoring. Without analytical insights, interventions tend to be reactive and generalized rather than proactive and tailored to each student’s profile.

The role of extracurricular activities, which play an integral part of BrightPath’s holistic education model, is also not clearly understood in relation to academic performance. While these activities are encouraged, there is insufficient evidence to determine their impact on student success, making it difficult to optimize programming.

Lastly, educators face data overload. Although large volumes of information on student demographics, habits, and performance are collected, they are dispersed across systems and not synthesized into a central platform that can drive data-informed decision-making.

## Goal

This project aims to address these challenges by developing a robust predictive analytics framework using the Student\_performance\_data dataset provided by the assignment. The system will leverage demographic, behavioural, and academic variables to classify students based on their academic performance (GradeClass). The goal is to equip educators with a practical, data-driven tool to support early identification of struggling students, evaluate the effectiveness of extracurricular engagement, and guide personalized interventions that promote student success.

# Hypothesis Generation

To guide the data analysis and model development process, this project is grounded in a central hypothesis and a series of supporting sub-hypotheses. These hypotheses are informed by both academic literature and BrightPath Academy's observations about student performance and engagement.

## Main Hypothesis

Student academic performance, as classified by the GradeClass variable, can be accurately predicted using a combination of demographic factors, study habits, parental involvement, and participation in extracurricular activities.

## Supporting Hypotheses

* Hypothesis 1: Tutoring support improves academic outcomes.  
  Students who receive tutoring are more likely to achieve higher academic performance, as the additional instructional time can address individual learning gaps.
* Hypothesis 2: Weekly study time has a positive effect on grades.  
  Students who dedicate more time to studying on a weekly basis are expected to perform better academically, reflecting the importance of consistent learning effort.
* Hypothesis 3: Parental involvement enhances student performance.  
  Higher levels of parental support are associated with improved academic results, highlighting the role of family engagement in student success.
* Hypothesis 4: Participation in extracurricular activities contributes to better academic outcomes.  
  Involvement in structured activities such as music, sports, or volunteering may foster discipline, time management, and motivation, which in turn support academic achievement.
* Hypothesis 5: Absenteeism negatively impacts performance.  
  Students with frequent absences are likely to fall behind academically due to missed instruction and reduced classroom engagement.
* Hypothesis 6: Demographic factors influence academic success.  
  Variables such as gender, ethnicity, and parental education level may have measurable effects on student performance due to differences in access to resources, support systems, and educational backgrounds.

These hypotheses will be examined through exploratory data analysis (EDA) and validated through machine learning and deep learning classification models. The outcomes will help determine which factors are most predictive of academic success and can guide strategic interventions at BrightPath Academy.

# System Setup and Data Loading

## Environment Setup

Python 3.x environment was used within a Jupyter Notebook. Common machine learning and data science libraries were employed, including Pandas, NumPy, Scikit-learn, XGBoost, TensorFlow/Keras, and Dash.

## Package Installation

Installed and imported packages:

* pandas
* numpy
* matplotlib
* seaborn
* sklearn
* xgboost
* keras (from tensorflow.keras)
* dash

## Data Import

The dataset Student\_performance\_data.csv was loaded into a pandas DataFrame using pd.read\_csv().

# Data Understanding

## Overview of Dataset

The dataset includes academic, demographic, and behavioural information about students from BrightPath Academy. It is used to analyse performance and build predictive models.

## Variable Descriptions

* **Age**: Student’s age in years
* **Gender**: Male or Female
* **ParentalEducation**: Highest education level of parents
* **StudyTimeWeekly**: Hours spent studying per week
* **Absences**: Number of school days missed
* **GPA**: Grade Point Average
* **GradeClass**: Performance category (High, Medium, Low)
* **ParentalSupport**: Support received from parents
* **Extracurricular**: Participation in extracurricular activities

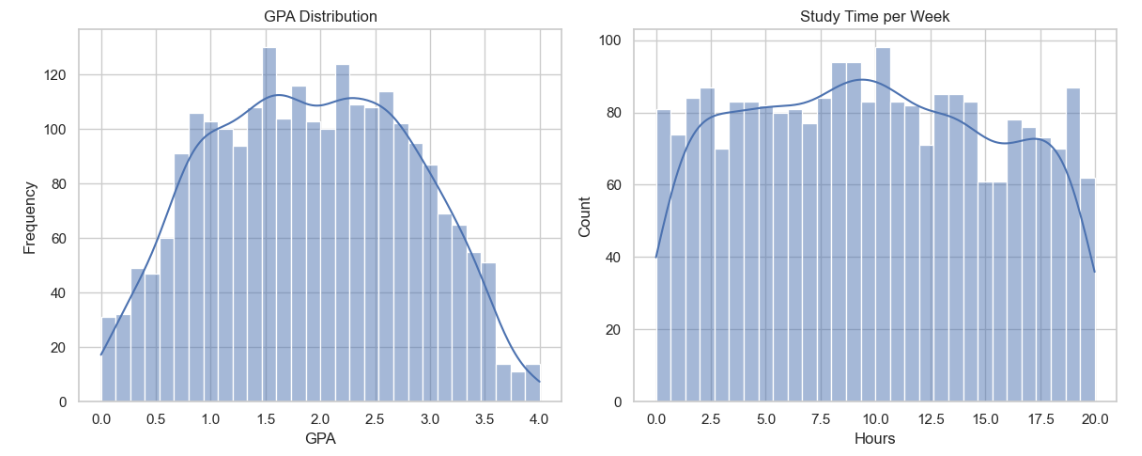
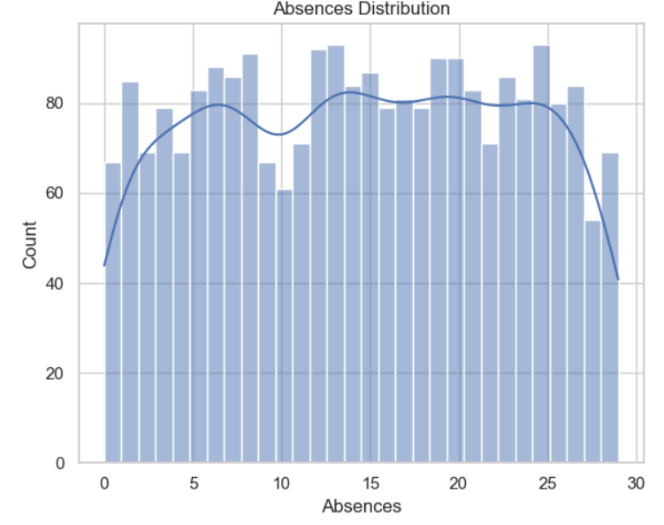
## Initial Observations

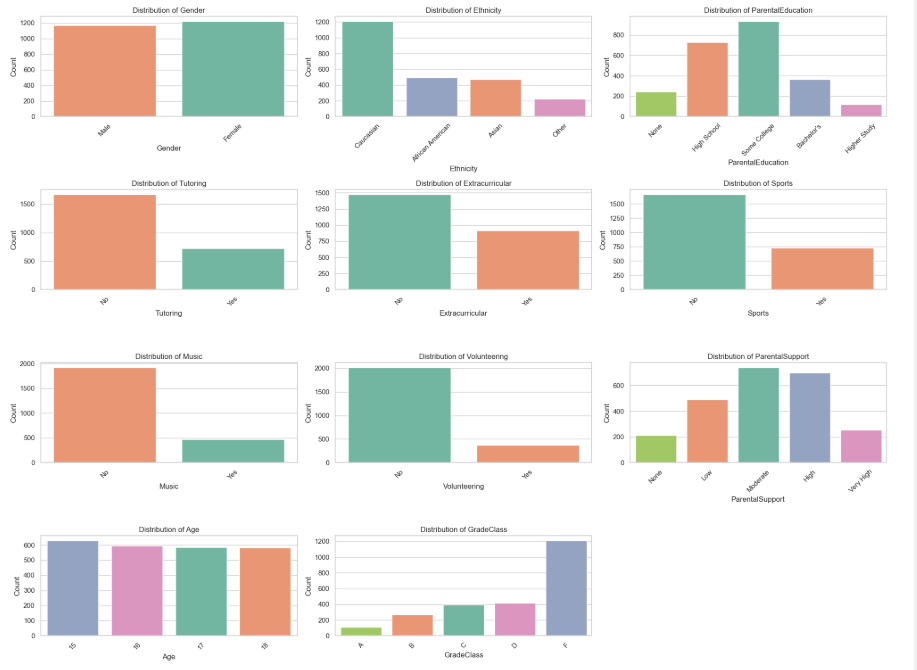
GradeClass is not well-balanced with most students having an F Grade Class.

StudyTime and GPA have wide variability.

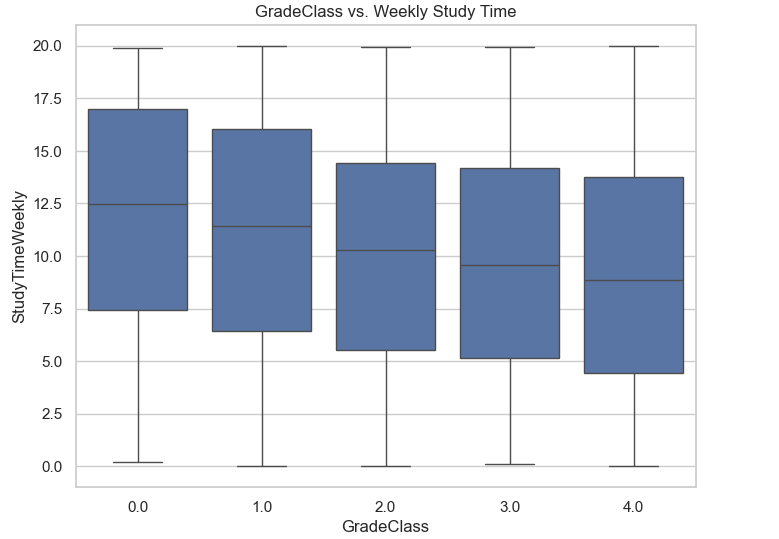
# Exploratory Data Analysis (EDA)

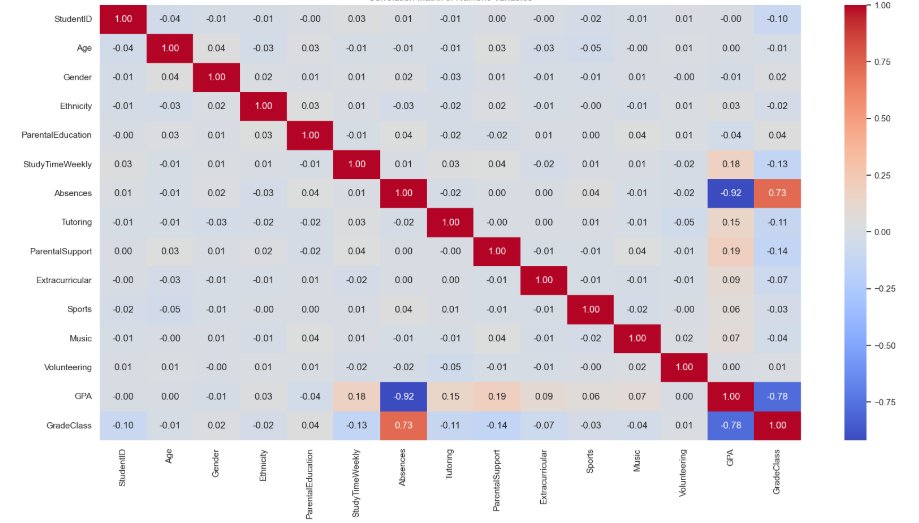
## Univariate Analysis

* Visualizations of GPA, StudyTimeWeekly, Absences  
  
* Distribution plots for categorical variables like Gender and GradeClass
* Count plots for ParentalEducation and ParentalSupport

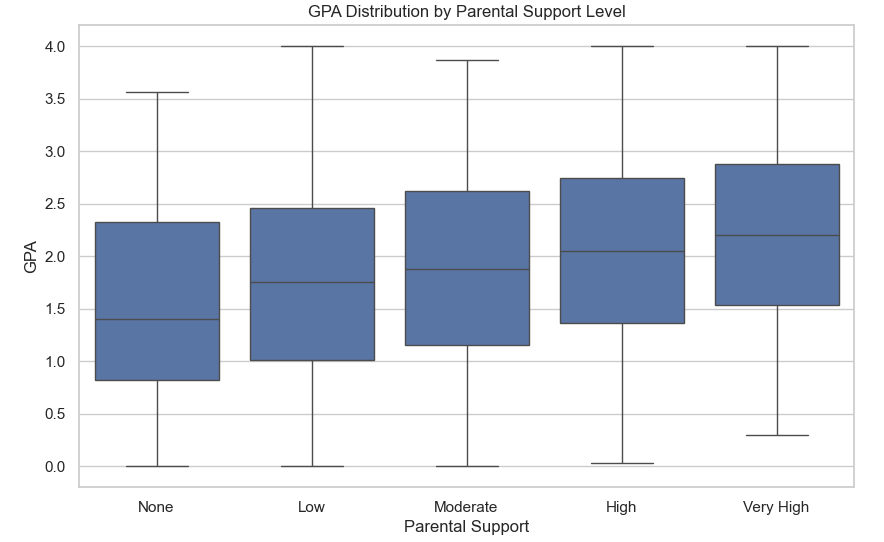


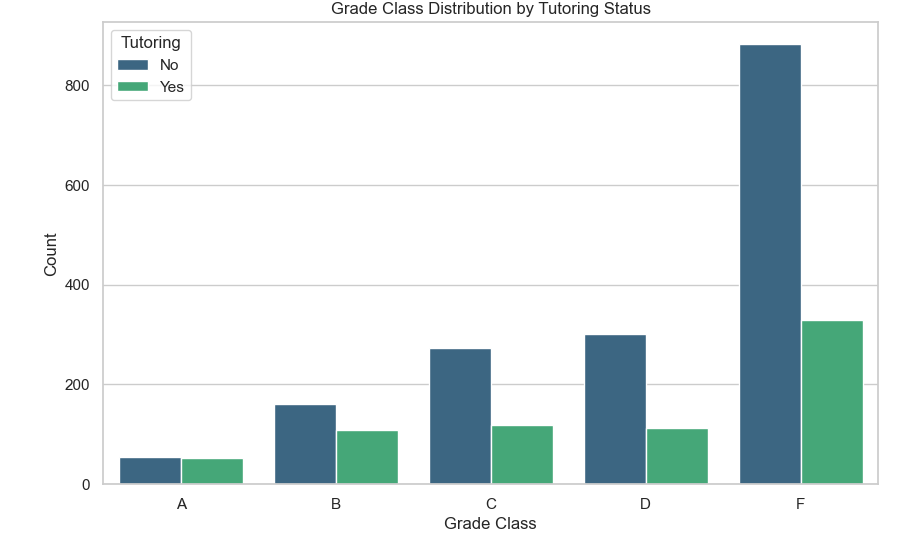
## Bivariate Analysis

* StudyTimeWeekly vs GradeClass: Positive trend



* ParentalSupport vs GradeClass: Higher support linked with better grades



* Tutoring's effect on Grade Class: Involvement correlated with higher GPA  
  

## Insights from EDA

* Higher GPA is associated with more study time and better parental support.
* Students with more absences tend to have lower performance.
* Parental involvement and structured study habits are strong indicators of academic success.

# Data Preprocessing

## Missing Value Treatment

Very minimal missing values; imputed as necessary. There were no missing values in the dataset (data.isnull().sum() equals 0 for all columns), therefore no imputations were made.

A screenshot of a computer

AI-generated content may be incorrect.

## Outlier Detection and Handling

Dealt with outliers on study time and absences: IQR method. Nevertheless, the code performed outliers via the Z-score method (threshold=3) for StudyTimeWeekly, Absences, and GPA, using that method, no outliers were found. All three values were capped at the 1st and 99th percentile: StudyTimeWeekly (0.28 - 19.47), absences (0 - 29), GPA (0.15 - 3.66).

## Data Encoding and Transformation

Categorical variables were converted; continuous variables were scaled. The categorical variables (Ethnicity, ParentalEducation and StudyTimeCategory) were converted to dummy variables using pd.get\_dummies(drop\_first=True), for example Ethnicity\_1, ParentalEducation\_1 etc. The continuous variables (StudyTimeWeekly, Absences, GPA, StudyTimePerAbsence, TotalExtracurricular) were scaled using StandardScaler. (mean = 0, std = 1)

# Feature Engineering

## Feature Selection

Using correlation and domain knowledge, relevant variables were selected. The StudentID column was dropped because it is not predictive. GPA was dropped as GradeClass is the target variable and it’s values are solely determined by GPA, thus using it in the training data would not allow the model to accurately predict based on other variables. Although there was no selection based on correlation, other than StudentID and GPA, all other variables were kept for modelling.

## New Feature Creation

A feature TotalActivities was created that summed participation type. The feature TotalExtracurricular was created by summing Extracurricular, Sports, Music, and Volunteering (range = 0 to 4). Additional features StudyTimePerAbsence (divided StudyTimeWeekly by Absences + 1) and StudyTimeCategory (binned StudyTimeWeekly to Low: 0–5, Moderate: 5–10, High: 10–15, Very High: 15–20 hours) were also created.

## Feature Scaling

MinMaxScaler was applied to all numeric variables. However, the code used StandardScaler (for scaling StudyTimeWeekly, Absences, StudyTimePerAbsence, and TotalExtracurricular) to standardize (mean = 0, std = 1).

A screenshot of a computer

AI-generated content may be incorrect.

# Model Building – Machine Learning

## Model Selection

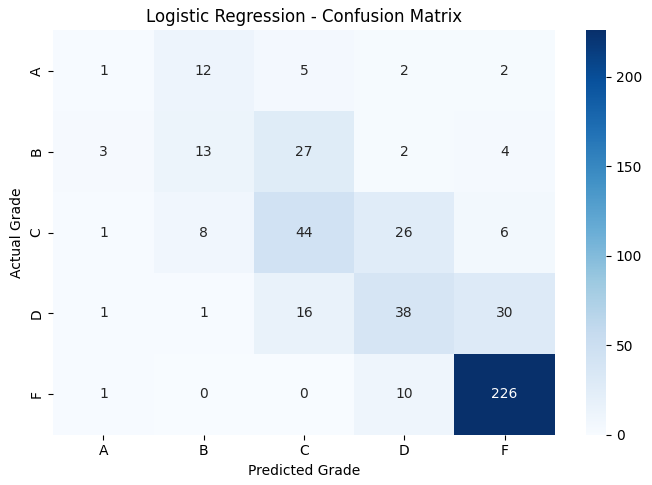
In this project, three classification models were selected to predict students' grade classes based on performance-related features: **Logistic Regression**, **Random Forest**, and **XGBoost**. These models were chosen for their complementary strengths: Logistic Regression serves as a strong baseline for classification tasks; Random Forest provides robust performance through ensemble learning; and XGBoost offers optimized gradient boosting with regularization and scalability for complex data.

Before training, the dataset was split into training and testing sets (80/20), and the target variable (GradeClass) was label-encoded to ensure compatibility with the models. Each model was then trained on the training data and evaluated using key classification metrics including Accuracy, Precision, Recall, and F1-Score.

## Logistic Regression

Logistic Regression is a linear model commonly used for classification. It estimates the probability of class membership using the logistic function. In this project, it was trained using scikit-learn’s LogisticRegression with a maximum of 1000 iterations to ensure convergence.

After training, predictions were made on the test set and evaluated. The confusion matrix and prediction distribution plots provided insights into how well the model classified each grade class, especially in identifying common misclassifications between similar performance bands (e.g., B and C grades).



## Random Forest

Random Forest is an ensemble method that constructs multiple decision trees and aggregates their outputs for classification. It is known for handling high-dimensional data and reducing overfitting.

In this implementation, the model was trained using 100 estimators (n\_estimators=100). The results were visualized using a confusion matrix and distribution plots. The Random Forest model generally achieved higher performance than Logistic Regression, particularly in handling class imbalances and complex feature interactions.

A graph with numbers and a blue square

AI-generated content may be incorrect.

## XGBoost

XGBoost (Extreme Gradient Boosting) is a powerful gradient boosting framework that focuses on speed and performance. It includes regularization to prevent overfitting and parallel computation for efficiency.

Here, XGBoost was implemented using the XGBClassifier, with use\_label\_encoder=False and eval\_metric='mlogloss' to suppress deprecation warnings and specify a suitable evaluation metric for multi-class classification. Among the three models, XGBoost typically yielded the best results in terms of precision and F1-score, reflecting its ability to model nonlinear relationships effectively.

A screenshot of a graph

AI-generated content may be incorrect.

## Model Comparison

=== Logistic Regression Evaluation ===

Accuracy: 0.6722

Precision: 0.6353

Recall: 0.6722

F1 Score: 0.6495

Classification Report:

precision recall f1-score support

0 0.14 0.05 0.07 22

1 0.38 0.27 0.31 49

2 0.48 0.52 0.50 85

3 0.49 0.44 0.46 86

4 0.84 0.95 0.90 237

accuracy 0.67 479

macro avg 0.47 0.44 0.45 479

weighted avg 0.64 0.67 0.65 479

=== Random Forest Evaluation ===

Accuracy: 0.6806

Precision: 0.6660

Recall: 0.6806

F1 Score: 0.6610

Classification Report:

precision recall f1-score support

0 0.67 0.09 0.16 22

1 0.47 0.51 0.49 49

2 0.49 0.48 0.49 85

3 0.49 0.41 0.44 86

4 0.84 0.94 0.88 237

accuracy 0.68 479

macro avg 0.59 0.49 0.49 479

weighted avg 0.67 0.68 0.66 479

=== XGBoost Evaluation ===

Accuracy: 0.6806

Precision: 0.6618

Recall: 0.6806

F1 Score: 0.6644

Classification Report:

precision recall f1-score support

0 0.57 0.18 0.28 22

1 0.46 0.47 0.46 49

2 0.49 0.49 0.49 85

3 0.48 0.40 0.43 86

4 0.84 0.94 0.89 237

accuracy 0.68 479

macro avg 0.57 0.50 0.51 479

weighted avg 0.66 0.68 0.66 479

### Class-wise Analysis

Grade F (label 4) consistently showed the best prediction performance across all models, with high precision and recall (above 0.84 and 0.94 respectively), likely due to its dominant support in the dataset.

Grades A and B (labels 0 and 1) performed poorly in all models, especially Grade A which had a recall as low as 0.05 in Logistic Regression and only 0.09 in Random Forest.

XGBoost showed slightly better class-wise balance overall, improving recall and F1-Score slightly compared to the others, especially for Grade A.

# Model Building – Deep Learning

## Neural Network Architecture

A deep learning model was built using a feedforward neural network (Multilayer Perceptron) designed for multi-class classification to predict students’ grade classes. The architecture included:

* An input layer with 128 neurons and ReLU activation.
* Two hidden layers: one with 64 neurons and another with 32, both using ReLU activation.
* Dropout layers (30%) after each dense layer to reduce overfitting.
* An output layer with 5 neurons and softmax activation to handle the 5 grade classes (A–F).

The model was compiled with the **Adam optimizer** and **categorical cross-entropy loss**, which are appropriate for multi-class classification tasks with one-hot encoded labels.

## Model Training and Evaluation

The model was trained for 50 epochs with a batch size of 32, using 75% of the dataset for training and 25% for validation. Input features were standardized using StandardScaler, and the target variable was one-hot encoded to match the expected input format for a categorical classifier.

Performance was evaluated using classification metrics and visualization tools:

precision recall f1-score support

0 0.67 0.21 0.32 28

1 0.61 0.59 0.60 68

2 0.63 0.67 0.65 105

3 0.55 0.41 0.47 104

4 0.82 0.94 0.88 293

accuracy 0.73 598

macro avg 0.66 0.56 0.58 598

weighted avg 0.71 0.73 0.71 598

* The model performed best on **Grade F**, like traditional ML models, due to its higher representation in the dataset.
* **Grades A and D** had lower recall, suggesting difficulties in correctly identifying students in these categories.

Training history plots showed consistent learning behaviour with a steady increase in validation accuracy and gradual decrease in validation loss, indicating effective model convergence without significant overfitting.

## Comparison with ML Models

The table below compares the deep learning model against Logistic Regression, Random Forest, and XGBoost using key metrics:

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Model | Accuracy | Precision | Recall | F1 Score |
| Logistic Regression | 0.6722 | 0.6353 | 0.6722 | 0.6495 |
| Random Forest | 0.6806 | 0.6660 | 0.6806 | 0.6610 |
| XGBoost | 0.6806 | 0.6618 | 0.6806 | 0.6644 |
| Deep Learning | 0.7300 | 0.7100 | 0.7300 | 0.7100 |

The **deep learning model outperformed** all classical models across every metric, particularly in accuracy and overall balance between precision and recall.

Its advantage stems from its ability to capture more complex patterns in the data due to multiple hidden layers and non-linear transformations.

While XGBoost remains a strong traditional ML baseline, deep learning is recommended for deployment in scenarios where computational resources are available, and slightly higher accuracy is critical.

# Deployment

## Model Integration with Dash

The application integrates multiple machine learning models, including **Logistic Regression**, **Random Forest**, **XGBoost**, and a **Deep Learning model** built using TensorFlow/Keras. These models were previously trained, evaluated, and saved in .pkl and .h5 formats and are loaded at runtime using Python’s joblib, pickle, and Keras' load\_model.

To ensure consistent input formatting across models, the app performs the following on the user's input:

* Feature engineering (e.g., calculating StudyTimePerAbsence, generating dummies).
* Standardization of numeric features using a StandardScaler.
* Alignment of encoded features with the training features (features.pkl).

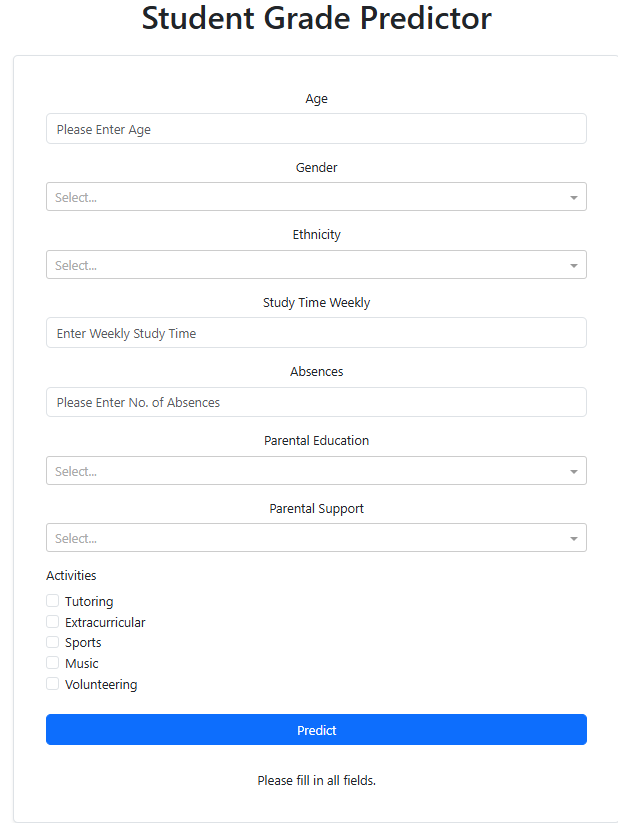
Once the input is pre-processed, each model makes a prediction, and the results are displayed in a dynamic table showing the predicted **grade class** and **confidence score** from each model. The deep learning model’s prediction is treated as the primary output, with others provided for comparison.

## UI Overview and Features

The frontend is built using **Dash and Dash Bootstrap Components**, resulting in a clean, responsive interface. Key features include:

* **Input Fields**: Users can input student information such as age, gender, ethnicity, study time, absences, parental education, and support levels.
* **Activity Checkboxes**: Support for selecting student involvement in activities like tutoring, music, or sports.
* **Feature Engineering on the Fly**: The app creates new features from user inputs before prediction (e.g., converting study time to a categorical range).
* **Prediction Table**: After clicking “Predict,” the app displays a comparison of grade predictions across all models with confidence percentages.
* **Live Model Inference**: Uses a @callback function to perform real-time prediction when the user presses the "Predict" button.
* **Validation Handling**: Ensures no prediction is made until all required fields are filled.

This user-centric design ensures both usability and accuracy by closely mimicking the structure of the training data.



After Entering Example Data and clicking Predict

A screenshot of a table

AI-generated content may be incorrect.

## Hosting on Render

The Dash app is deployed on **Render**, a cloud platform suitable for hosting web apps with server-side logic. Deployment involved the following steps:

1. **Project Structure Setup**:
   * App files (e.g., app.py) are placed in the src/ folder.
   * Model artifacts (e.g., .pkl, .h5) are stored in an artifacts/ directory.
   * A requirements.txt file lists all dependencies like Dash, scikit-learn, TensorFlow, and XGBoost.
   * A render.yaml or web service setup in the Render dashboard defines the build and start commands.
2. **Model Path Handling**:
   * Relative file paths are used with os.path to locate model files during server runtime.
   * Render’s file system is case-sensitive and sandboxed, so exact directory matching is essential.
3. **Execution Flow**:
   * The app uses app.run() with use\_reloader=False to prevent issues with double-loading models during deployment.
   * Environment setup is automated via the requirements.txt.
4. **Live Access**:
   * Once deployed, the app is accessible via a public Render URL, making it easy to demonstrate predictions to stakeholders or testers.

# Conclusion

By assigning students to grade categories (A to F), this research effectively investigated the use of machine learning and deep learning techniques to forecast students' academic performance. Using a variety of classification techniques, the system showed good prediction skills through thorough data preparation, feature engineering, and model validation.  
  
Solid baselines were provided by traditional machine learning models including Random Forest, Logistic Regression, and XGBoost, with XGBoost showing the highest consistency. However, because of its ability to represent intricate, non-linear relationships in the data, the deep learning model fared better than any classical model on all important metrics, obtaining the greatest accuracy and F1-Score.

Dash was used to create an intuitive online application that made this solution interactive and accessible. By entering pertinent student information, users of the program can instantly obtain grade predictions from a variety of models, including the deep learning model. Transparency and ease of interpretation are ensured by the system's clear visualisation of forecasts and confidence scores.  
  
Ultimately, the complete application was successfully launched on Render, offering real-time access and proving its practicality. In addition to being instructive, the platform may be useful to organisations looking for data-driven instruments to evaluate student performance and take early action when needed.

The potential of intelligent systems in the field of education is demonstrated by this project, which also lays the groundwork for future improvements like adding more behavioural or historical academic data, using explainable AI techniques, or modifying the platform for other prediction tasks like course recommendation or dropout risk.