**BrightPath Academy - Student Performance Analysis**

**Problem Statement**

BrightPath Academy aims to improve student performance by identifying the key factors that influence academic success. Using historical student data, we intend to build predictive models that can help the academy proactively support students at risk of underperforming.

**Hypothesis**

We hypothesise that factors such as attendance and socioeconomic indicators significantly impact a student's performance. By analysing and modelling these variables, we can predict whether a student is likely to pass or fail.

**Summary of Methodology**

**Data Loading and Initial Checks**

We began with importing and setting up the system, adding various tools for splitting the data and building and evaluating models, so that it could handle the following:

  - The student performance data tables

  - Numerical operations

  - visualising the system

  - Saving and loading the models

  - Interacting with the operating system

  - Setting the number of threads for parallel processing

  - Suppressing unwanted warning messages when loading the student performance dataset

Once an environment was set up, the dataset was loaded using the 'Student\_performance\_data.csv' file. Initial checks were then performed to understand the structure and contents of the data. This included checking for null values, data types, and basic statistics.

**Exploratory Data Analysis (EDA)**

In this phase, we explored data distributions, correlations, and relationships between variables. Visualisations and descriptive statistics were used to understand trends and identify significant features.

**Univariate Analysis**

**- Grade class** – we looked at the number of students in each class of seeing that most of them fell in class 4.

**- Study time weekly** – there were no outliers or anyone who spent more than 20 hours studying. And the study time had a decently normal distribution.

- **Univariate analyses** of all the features to see the scale and to help with scaling and outlier treatment.

**Bivariate Analysis**

-**Parental Support vs GPA**: Positive trend — GPA increases with higher support.

- **StudyTime vs GPA**: Moderate positive correlation (**r ≈ 0.54**, **p < 0.001**).

- **Absences vs GPA**: Moderate negative correlation (**r ≈ -0.39**, **p < 0.001**).

- **Ethnicity and Participation Variables**: Boxplots and histograms helped evaluate categorical variable influence on GPA.

Correlation metrics were embedded in graphs to quantitatively describe the strength and significance of relationships.

**Missing Value Treatment**

- Columns with minimal missing values were handled using imputation:

  - **Numerical columns** such as GPA: Filled with **median**.

  - **Categorical columns** such as ParentalSupport: Filled with **mode**.

This preserved the underlying distribution without biasing the data.

**Sigmoid Function Simulation (Logistic Regression Illustration)**

To build conceptual bridges with the classification phase, we applied a **simulated logistic regression** model using the sigmoid function:

- Created a binary target variable: GPA\_High = 1 if GPA ≥ 3.5, else 0.

- Simulated a logistic function using:

LinearCombo = 0.35 \* StudyTimeWeekly - 2.5

PredictedProb = 1 / (1 + e^(-LinearCombo))

**Feature Engineering**

Relevant features were created and transformed to improve the predictive power of the dataset. This included encoding categorical variables and scaling numerical values where necessary.

**Model Building**

**Part 1: Traditional Machine Learning Models**

Multiple classification models were used to predict student performance, including **Logistic Regression**, **Random Forest**, and **XGBoost**. Each model was trained on the pre-processed dataset and evaluated using appropriate metrics.

**Part 2: Artificial Neural Network (ANN) Model**

To further enhance the classification performance of our student performance prediction system, we implemented an Artificial Neural Network (ANN). This deep learning approach allowed us to capture more complex patterns in the data that traditional models may overlook.

**Data Preparation**

Before training the ANN, the categorical target labels were encoded using a label encoder. The encoded labels were then one-hot encoded to suit the multiclass classification problem, with five distinct output categories. The dataset was split into training and testing sets using an 80-20 split, ensuring that model performance could be evaluated on unseen data.

**Model Architecture**

The ANN was built using a sequential model structure consisting of the following layers:

- **Input Layer**: A dense layer with 64 neurons and ReLU activation, corresponding to the number of input features.

- **Hidden Layers**:

  - A dropout layer with a dropout rate of 0.3 was applied after the first dense layer to prevent overfitting.

  - A second dense layer with 32 neurons and ReLU activation was added to further capture patterns.

  - A second dropout layer with a dropout rate of 0.2 followed.

- **Output Layer**: A dense layer with 5 output neurons (one for each class), using softmax activation for multiclass classification.

**Compilation and Training**

The model was compiled using the **Adam optimiser**, and the **categorical cross-entropy** loss function was used since it was a multiclass problem. The model was trained for **50 epochs** with a **batch size of 32**, and validation was performed on the test set to monitor overfitting and generalisation.

**Performance Evaluation**

- The model achieved a **high training accuracy**, demonstrating its ability to learn from the data.

- The **testing accuracy** also indicated strong generalisation performance, suggesting that the ANN effectively captured the underlying patterns without overfitting.

**Visualization**

To visually interpret model learning, **accuracy plots** for both training and validation sets were generated across the 50 epochs. These plots provided a clear view of model convergence and stability during training.

**Evaluation Metrics**

Model performance was evaluated using metrics such as **accuracy**, **precision**, **recall**, **F1-score**, and **ROC-AUC**. This helped us determine which model was the most effective in identifying students who were likely to fail.