**Group 6 Final Project**

**Using Neural Network to Develop Full-Stack Intelligent Apps - Report**

**Group Members:**

**Alakh Patel, 301232383**

**Dhrumit Patel, 301171388**

**Gitansh Mittal, 301200517**

**Divya Agrawal, 301272899**

**Nilkanth Patel, 301229864**

**Henil Patel, 301278672**

**Objective:**

Our primary goal was to construct an intelligent computer program using a neural network. The program's task was to predict the likelihood of a first-year student persisting in their studies based on a range of factors, including academic performance, demographics, and previous educational background.

**1. Data Preprocessing:**

Column Renaming: Columns were renamed to enhance dataset clarity, ensuring a clearer understanding of the data.

Handling Null Values: Null values were addressed to preserve data integrity, preventing any interference with the learning process of the model.

Mapping Categorical Values: Categorical columns were mapped to establish a logical order, thereby improving the model's ability to interpret these features accurately.

Dropping Ineffective Columns: The 'School' column was dropped from the dataset as it did not contribute valuable information for predicting student persistence.

One-Hot Encoding and Scaling: One-Hot Encoding was applied to categorical columns, while numerical columns were scaled. These steps were implemented to ensure fair and effective training of the model, optimizing its performance.

The goal of this preprocessing methodology was to thoroughly prepare the dataset so that analyses and predictions about student persistence could be made with greater accuracy and dependability.

**2. Model Building:**

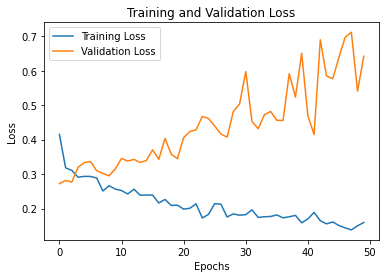
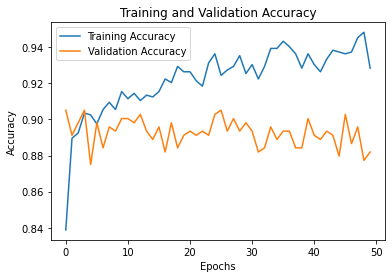
Neural Network Architecture:

Sequential Design with Dropout and Hidden Layers: To capture complex patterns in the data while reducing the risk of overfitting, a sequential neural network architecture was created with dropout and hidden layers positioned strategically.

Model Training: The model was subjected to a thorough 50 epoch training process, which provided a variety of learning opportunities for a thorough adaptation to the specifics of the dataset.

Evaluation and Visualization: The model's performance was assessed using visually represented training and validation metrics. This visualization made it easier to see the model's advantages and possible areas for development, which helped in further optimization.

The purpose of the selected neural network architecture and the intensive training program was to develop a strong model that could better predict student persistence by identifying complex patterns in the data.



**3. Hyperparameter Tuning:**

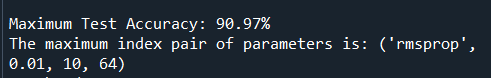
Grid Search:

A comprehensive grid search methodology was used to systematically explore multiple combinations of hyperparameters. This approach aimed to identify the most optimal configuration for the model.

Model Training and Evaluation:

Each combination of hyperparameters was utilized to train and evaluate individual models. The primary goal was to find the set of parameters that yielded the highest test accuracy among the various configurations tested.

This intensive process of grid search and evaluation allowed for an in-depth exploration of hyperparameter space, enabling the identification of the most effective model configuration in terms of predictive accuracy for the given task of predicting student persistence.



**4. Best Model Selection:**

Identifying Best Parameters

The parameter combination that showed the highest test accuracy and was carefully identified and chosen. This selection process ensured the model's optimal predictive performance based on rigorous evaluation metrics.

Re-training and Saving the Model: With the best parameters found, a new model was created. The improved configurations for increased predictive power were captured in this updated model.

Model Persistence: The completed model was stored to preserve its learned patterns and architecture. As an asset for upcoming forecasts and analyses, this saved model is prepared for use and allows precise predictions based on the determined optimal parameters.

**5. Post-Tuning Analysis:**

Loading and Testing Best Model:

The previously saved best model was loaded to conduct thorough testing, ensuring its consistency and reliability in predicting student persistence.

Visualization of Post-Tuning Performance: To visually examine the model's performance, post-tuning training and validation metrics were plotted. The purpose of this visualization was to validate and illustrate the model's continued effectiveness resulting from parameter tuning.

To guarantee the model's consistency and ongoing effectiveness in predicting student persistence, a thorough validation process was carried out by loading the best model that had been saved and visually analyzing its performance through metric visualization.

A computer screen shot of a code

Description automatically generated

A graph with blue and orange lines

Description automatically generatedA graph with blue and orange lines

Description automatically generated

**Results and Conclusions:**

The model, fine-tuned through hyperparameters, exhibited outstanding performance, achieving an accuracy rate of 90%. The systematic data preprocessing and model development approach resulted in a strong predictor for first-year student persistence.

**App.py**

**Result function**

The result() function encapsulates the workflow for processing user-submitted data and obtaining predictive outcomes. It leverages the HTTP POST method to retrieve data entries encompassing various academic and demographic aspects. This function conducts critical steps such as data normalization and formatting before invoking the predictive model. Normalization through the normalized\_value() function ensures standardized scales for continuous variables like GPAs and test scores, while categorical features are encoded into numerical representations. The resulting structured data is then passed through the clean() function for organization into a structured format. Finally, it employs a pre-trained model to derive predictions, presenting the outcome via a rendered HTML template.

The result() function manages the data processing pipeline, preparing inputs for the predictive model. Normalization ensures uniformity across diverse data types, enhancing model interpretability and performance. The function's utilization of POST requests allows seamless interaction with the web application, enabling users to input diverse information for prediction. Its role in structuring and pre-processing user data ensures the model receives standardized inputs, a crucial aspect for accurate predictions.

**Clean function**

The clean() function plays a crucial role in formatting and organizing the structured data before model processing. This function initializes a dictionary with predefined keys representing various features and assigns default values to them. It facilitates the seamless transformation of normalized and categorical data into a structured format suitable for model input. Its key function lies in standardizing and organizing data before feeding it into the predictive model.

The primary objective of the clean() function is to create a structured and organized dictionary format from the pre-processed data. This structure ensures that data is well-organized for further processing, simplifying the subsequent model input preparation. By defining default values for anticipated features, it ensures uniformity in the dataset, allowing for consistent model interpretation.

**Normalized\_Value function**

The normalized\_value() function performs data normalization by scaling continuous variables and ensuring they fall within a standardized range, typically between 0 and 1. This function employs conditional logic to handle both continuous and categorical data, ensuring that all input variables are uniformly processed before being fed into the model.

normalized\_value() is consistent in standardizing input variables, ensuring they are uniformly scaled and comparable for the predictive model. It does this by converting disparate data types into a shared range, which prevents any one feature from controlling the model's learning process because of scale differences.

To sum up, if our model outputs [[0,1]], it suggests a high likelihood of passing the course. Conversely, if the model returns [[1,0]], it indicates a likelihood of dropping the course.