Developing a Neural Network for Predicting Final Exam Scores Based on Study Habits and Attendance

ICT3212 – INTRODUCTION TO INTELLIGENT SYSTEMS

Group Name: Unified Explorers

Name	Registration Number	Index Number
R.Priyatharsan	ICT/2021/092	5248
T.Nivethiha	ICT/2021/081	5237
S.Sangavan	ICT/2021/103	5259
S.Kamsala	ICT/2021/052	5213
J.Thayanithi,	ICT/2021/050	5211

Mentored By: **H. L. A. R. Lavan Deepasara**

Faculty of Applied Sciences

Rajarata University of Sri Lanka

2020/2021 Batch

Table of Contents

1. Introduction

Academic performance is influenced by a variety of factors, including study hours, subject credits, attendance, and assignment marks. Monitoring these factors is essential for identifying students who may require additional support to succeed academically. Predicting final exam outcomes in advance can help educators provide timely interventions and improve overall student performance.

This project aims to predict final exam results using a neural network model. The model analyzes factors such as study hours, attendance, assignment marks, subject credits, and past results to identify patterns in student performance. Artificial Intelligence is applied to detect patterns and make predictions. Neural networks are effective because they can capture complex relationships in the information.

For this project, we selected the Department of Computing, Faculty of Applied Sciences, Rajarata University of Sri Lanka, as the target population to implement and test this predictive system. The main goal of this project is to help students succeed, lower failure rates, and support decisions with information.

2. Problem Statement

One of the main issues is that the organization cannot easily identify students who might perform poorly in final exams based on their study habits, attendance, and assignment marks. Although information on weekly study hours, attendance, GPA, and assignment marks are collected, it is not analyzed in a systematic way to predict student outcomes or guide support services. Predicting exam results manually is time-consuming and often inaccurate, which makes it difficult to provide timely and personalized help. Basic methods of analyzing data, such as looking at averages or simple relationships between factors, are not always able to show how different student behaviors interact to affect exam performance. This highlights the need for a data-driven predictive tool that can consider multiple factors and provide reliable estimates of final exam results, enabling early intervention and better support for students.

3. Data Collection

The dataset for this project was collected from the undergraduates of the Department of Computing, Rajarata University of Sri Lanka. Therefore, our target population is 384 Specific population was selected to reduce biasness, to keep consistent subject structures and evaluation methods. We employed a convenience random sampling method to select participants. This approach was chosen because students were readily accessible to us, making data collection more practical and efficient. At the same time, the randomness in participation ensured that the collected data reflected varying academic abilities, study habits, and engagement levels within the department, thereby reducing bias.

To ensure that the sample size was statistically adequate, we applied a sample size calculation using ***Figure:01*** formula. The minimum required sample size = 192.29 ~ 193 Undergraduates.

$$n = \frac{N * Z^2 * p\,(1 - p)}{e^2 * (N - 1) + Z^2 * p\,(1 - p)}$$

Figure:01- Formula

n - Required sample size, N - total population (384), Z - Z- score based on confidence level 95% (1.96), p - Standard of deviation (0.5), e - Margin of error (5%)

Next, Data was collected using google form, which served as a convenient and cost-effective tool for gathering responses. Google Form was selected because it allowed structured question formats, automated data storage, and easy distribution among the student population. This method also encouraged voluntary participation, which helped in obtaining more genuine responses from students. Our google form link is: https://forms.gle/s1b9Aoxwar6XzgJm6. Using this google form we collected E-mail address, Faculty name, Department, Year of study, Number of course in last semester, Course Name, Course credit, Weekly study hours spent to study for each subject, Attendance percentage, Assignment marks and Final exam results. The Google Form was designed in such a way that all questions were made compulsory for the respondents. As a result, the dataset obtained was completed and more reduced the need for extensive handling of null values.

3.1 Data Preprocessing

Once the raw data was collected, data preprocessing was performed to clean and prepare the dataset for model training. Initially, only the relevant columns were selected to predicting final results. This ensured that irrelevant or redundant features were excluded, improving model focus and reducing computational overhead.

1. Course credit: Course credit represents the weight or importance of a course. A course with higher credit usually demands more effort and has a larger impact on the final grade. Including it helps the model understand the significance of performance in different courses.

2. Attendance percentage: High attendance often correlates with better understanding of the subject, more engagement, and better academic performance. Poor attendance may signal disengagement or lack of understanding. It's a strong behavioral indicator.

3. Weekly study hours: The amount of time students spends studying directly influences their performance. More study hours often lead to better preparation and higher achievement, though it may vary by individual learning efficiency.

4. Assignment Marks: Assignment performance reflects a student's understanding and engagement with the subject throughout the course. It gives a continuous assessment perspective and is usually a good predictor of final exam performance.

5. Final Results: It serves as the key measure of academic achievement.

To improve readability and avoid long column names, the dataset column names were renamed to shorter, more descriptive forms. The attendance values, originally expressed in ranges, were standardized by converting each range into its mean value. Similarly, the final exam results contained detailed grades such as A+, A-, B+, etc. These were simplified into five broader categories (A, B, C, D, and E). This step reduced noise and class imbalance while maintaining meaningful distinctions in performance levels, making the prediction task more manageable and consistent. Since the target variable was categorical, a label encoding was applied, which converted grades (A–E) into numeric values (A=0, B=1, C=2, D=3, E=4). This was essential because machine learning algorithms require numerical input to perform classification tasks.

After encoding and cleaning, we applied standard scalar to normalize the feature values. This transformation adjusts each numerical feature to have a mean of zero and a standard deviation of one. The reason for this step is that the raw values of features exist on very different scales. Then the dataset was split into training and testing subsets. We allocated 80% of the data for training and 20% for testing, with stratification applied to preserve the proportion of grade categories across both sets. This ensured that the model was trained on a representative distribution of the data while also being evaluated on unseen data, allowing a fair assessment of its predictive performance.

4. Model Design

The predictive model is implemented using a neural network built with the TensorFlow Keras library. The input layer consists of nodes equal to the number of features in the dataset (study hours, attendance, assignment scores, subject credits), allowing the model to process all relevant student information.

The network architecture includes three hidden layers:

- **Hidden Layer 1:** 128 neurons with ReLU activation, followed by a dropout layer (initially set to 0.3).
- **Hidden Layer 2:** 64 neurons with ReLU activation, followed by a dropout layer (initially set to 0.3).
- **Hidden Layer 3:** 32 neurons with ReLU activation, followed by a dropout layer (initially set to 0.2).

The output layer contains 5 neurons with softmax activation, corresponding to the five performance categories for student predictions. The model is compiled using the Adam optimizer with an initial learning rate of 0.001, and we used sparse categorical cross-entropy as the loss function because our target labels were label-encoded as integers (A=0, B=1, etc.). This loss function is suitable in such cases, as it directly works with integer labels instead of requiring one-hot encoded vectors. Accuracy is used as the primary evaluation metric.

To enhance training robustness, callbacks are employed:

- EarlyStopping (monitors validation loss, patience = 10) prevents overfitting by halting training when no improvement is observed.
- ModelCheckpoint saves the model achieving the highest validation accuracy.

The model is trained for up to 50 epochs with a batch size of 8, using 20% of the training set for validation. This architecture enables the network to learn complex relationships in student data and predict final exam performance effectively.

4.1 Hyperparameter Optimization

To improve predictive performance, hyperparameter optimization was conducted using both Grid Search and Bayesian Optimization. Grid Search systematically explored discrete combinations of batch size, epochs, dropout rates, and learning rate. Specifically, epochs [30, 50], batch sizes [8, 16], learning rates [0.001, 0.0005], and dropout rates [0.0, 0.2, 0.3] for each hidden layer were tested. The best combination identified was batch size = 8, epochs = 50, dropout rates = [0.0, 0.2, 0.2] for layers 1-3, and learning rate = 0.001.

Bayesian Optimization was then applied to fine-tune dropout rates and learning rate within continuous ranges. Dropout rates for the three hidden layers were searched in the range 0.0 – 0.3 with a step of 0.1, and the learning rate was optimized in the range 0.001 – 0.01 using log-scale sampling. By iteratively exploring the hyperparameter space based on previous evaluations, Bayesian Optimization identified the optimal configuration of dropout rates [0.1, 0.1, 0.0] and learning rate = 0.0013386608968322061.

Overall, the combination of Grid Search and Bayesian Optimization allowed exhaustive testing for discrete parameters and efficient fine - tuning for continuous ones, leading to more stable learning, reduced overfitting, and improved generalization for predicting student performance.

4.2 Documentation in Fine - tuning

During the model tuning process, we systematically recorded all hyperparameter adjustments and their impact on model performance to ensure reproducibility and clarity for future reference.

Hyperparameters Tuned:

- Dropout rates: 0.0 – 0.3 (step size 0.1 for Bayesian, selected discrete values [0.0, 0.2, 0.3] for Grid Search)
- Learning rate: 0.001 – 0.01 (log-scale for Bayesian), [0.001, 0.0005] for Grid Search
- Batch size: [8, 16]
- Epochs: [30, 50]

Tuning Methods:

1. Grid Search: Used to test discrete combinations of batch size, epochs, dropout rates, and learning rate.
2. Bayesian Optimization: Applied to efficiently fine-tune continuous hyperparameters (dropout rates and learning rate) based on probabilistic modeling of prior performance.

Results of Tuning Methods:

1. **Grid Search:**
   - Best parameters: {'batch_size': 8, 'epochs': 50, 'model_drop1': 0.0, 'model_drop2': 0.2, 'model_drop3': 0.2, 'model_learning_rate': 0.001}
   - Best CV accuracy: 0.8539325842696629

Figure:02, Figure:03 shows the results between the base line model and tuned model using Grid Search

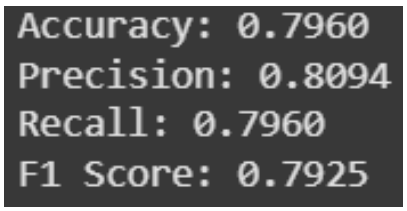


Figure:02

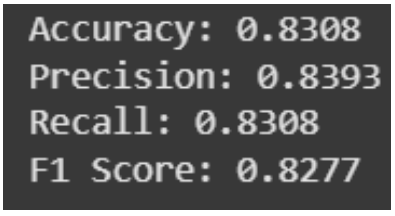


Figure:03

2. **Bayesian Optimization:**

Best hyperparameters found:
- Dropout 1: 0.1
- Dropout 2: 0.1
- Dropout 3: 0.0
- Learning Rate: 0.0013386608968322061

Figure:02, Figure:04 shows the results between the base line model and tuned model using Bayesian Optimization

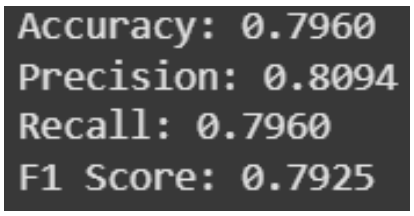


Figure:02

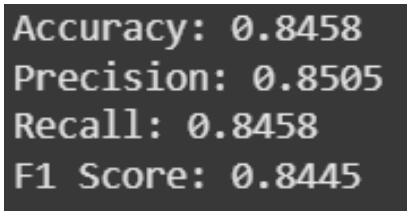


Figure:04

Rationale

The decisions behind tuning specific hyperparameters were based on their impact on model learning and generalization:

- **Dropout Rates:** Tuned to prevent overfitting. Lower dropout for the last layer was chosen because excessive dropout in deeper layers can reduce learning capacity.
- **Learning Rate:** Tuned over a log-scale to balance convergence speed and training stability. A learning rate that is too high can cause unstable training, while too low slows down learning.
- **Batch Size and Epochs:** Grid Search was applied since these are discrete hyperparameters with limited reasonable values. Smaller batch sizes allow finer updates to weights, and higher epochs ensure sufficient training.
- **Tuning Methods:** Grid Search was suitable for discrete parameters to exhaustively test combinations, while Bayesian Optimization efficiently explored continuous ranges and leveraged prior evaluation results to converge to optimal settings faster.
- **Final Selection:** The combination of these tuned parameters improved predictive accuracy and reduced overfitting, ensuring the model generalizes well to unseen student data.

5. Results & Evaluation

The dataset was divided into training and testing sets using an 80:20 ratio, with stratification applied to ensure class balance across both subsets. This allowed the model to learn from a representative distribution of student outcomes while still leaving enough unseen data for fair testing. After training, the model was evaluated on the test set to measure its ability to generalize beyond the training data. Learning curves were analyzed during this process, and they showed a consistent trend between training and validation performance. This stability confirmed that the model was neither overfitting nor underfitting, and that the training procedure had successfully captured the underlying patterns in the dataset.

To improve the baseline model, we experimented with different hyperparameter optimization techniques. Both Grid Search and Bayesian Optimization were applied to identify the best parameter combinations. While Grid Search provided a systematic approach to exhaustively explore predefined parameter ranges, it was computationally more expensive and less efficient when dealing with multiple hyperparameters. In contrast, Bayesian Optimization applied a probabilistic model to guide the search more intelligently, focusing on promising regions of the parameter space. From this process, Bayesian Optimization produced the highest accuracy, and therefore the hyperparameter set obtained from it was selected for building the final fine - tuned model. This ensured that our chosen model configuration represented the most effective balance between performance and computational efficiency.

The fine - tuned model was then evaluated using multiple classification metrics. ***Figure:04*** shows multiple classification of metrics.

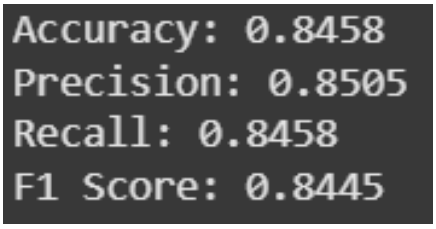


Figure:04 - Fine - tuned model Evaluation metrics

It achieved an accuracy of 84.58%, indicating that the majority of student final grades were predicted correctly. The precision score of 85.05% demonstrated that when the model predicted a particular student outcome, it was correct most of the time, minimizing false positives. The recall score of 84.58% confirmed that the model successfully captured most actual positive cases, avoiding high false-negative rates. Finally, the F1-score of 84.45%, which balances both

precision and recall, highlighted the model's strong overall performance across both dimensions. The confusion matrix visualization reinforced these results by providing a clear breakdown of correct and incorrect classifications, with relatively few misclassifications observed. ***Figure:05*** shows the confusion matrix

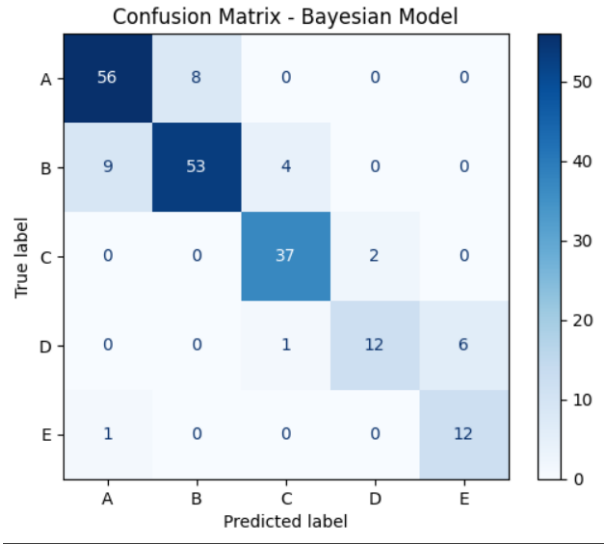


Figure:05 – Confusion Matrix

Beyond numerical metrics, interpretability and visualization techniques were applied to better understand the model's behavior. The confusion matrix allowed us to identify areas where the model performed strongly and where it faced challenges. Learning curves further validated that the model's generalization capability remained stable across epochs.

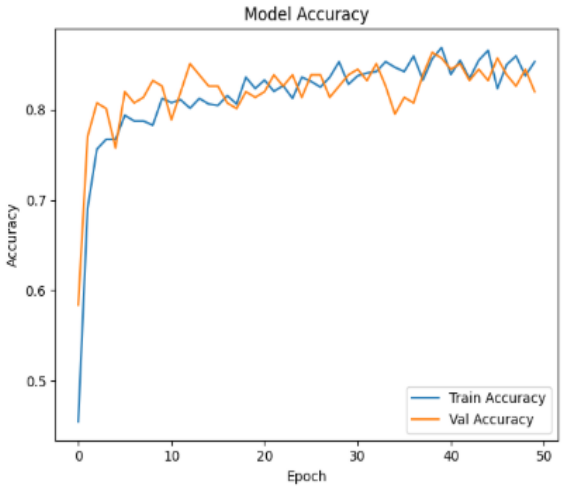


Figure:06

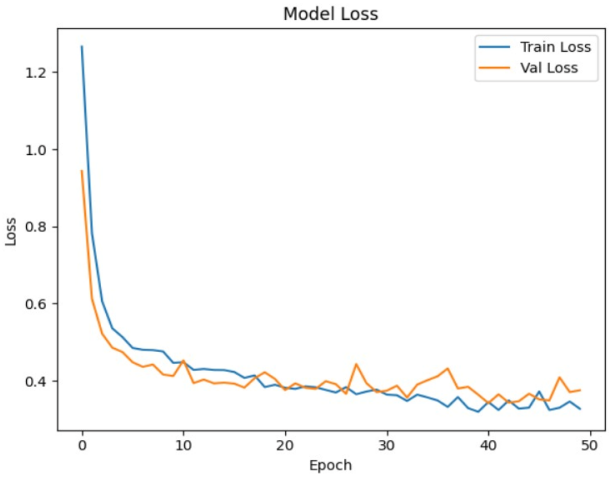


Figure:07

Figure:06 shows the model accuracy. How well a machine learning model learns over 50 training epochs. At the start, both training and validation accuracy increase quickly, meaning the model is learning patterns from the data. In the middle, the lines go up and down a little but stay close together, which shows the model is working well on both training and unseen data without overfitting. After about 20 epochs, both accuracies level off around 85% to 90%, so the model has reached its best performance. Since the validation accuracy stays close to the training accuracy, the model is generalizing well.

Figure:07 shows the model loss. How the model loss, changes over 50 training epochs. At the start, both training and validation loss are high, but they drop quickly as the model begins to learn. In the middle, the losses keep going down and become close to each other, meaning the model is learning well on both training and unseen data. After about 20 rounds, the losses level off around 0.35 to 0.4, showing the model has reached stable performance. Since the validation loss stays close to the training loss and does not rise, the model is not overfitting and is generalizing well. Training longer than 20 rounds does not bring much improvement.

Additionally, feature importance analysis suggested that factors such as attendance, prior academic performance, and student engagement had the highest influence on predictions. These insights aligned with educational theory, where consistent participation and strong past performance are key indicators of student success. Overall, the combination of robust hyperparameter tuning, strong evaluation metrics, and meaningful interpretability confirmed that the final fine-tuned model not only performs effectively but also provides valuable insights for supporting educators in decision-making and targeted student interventions.

6. Discussion & Reflection

Through this project, we gained a clearer understanding of how neural networks can be applied to predict student performance. By analysing academic and behavioural data such as study hours, attendance, course credits, and assignment marks, our model was able to identify useful patterns. We found that attendance and study hours were strong indicators of higher grades, showing the importance of consistent study habits and active participation. Assignment marks also helped distinguish top performers from weaker ones. However, predicting mid-range grades like B and C was more difficult, since students in this group often showed similar habits but achieved slightly different outcomes.

During development, we faced several challenges. Exam grades had to be grouped into five categories (A–E) so the model could process them effectively. Attendance records also needed to be cleaned and normalized due to missing or inconsistent values. Overfitting was another challenge, where the model performed well on training data but less effectively on test data. We addressed this using dropout layers, early stopping, and normalization. Model optimization was also important - Bayesian optimization produced the best accuracy of around 85%, while grid search gave slightly lower accuracy but required more training time.

What stood out to us was the potential of such models to support education in practical ways. With early predictions, teachers could identify students at risk of underperforming and provide targeted support such as mentoring or tutoring. Since the model also provides prediction probabilities, students themselves could benefit from personalized feedback about areas they need to improve.

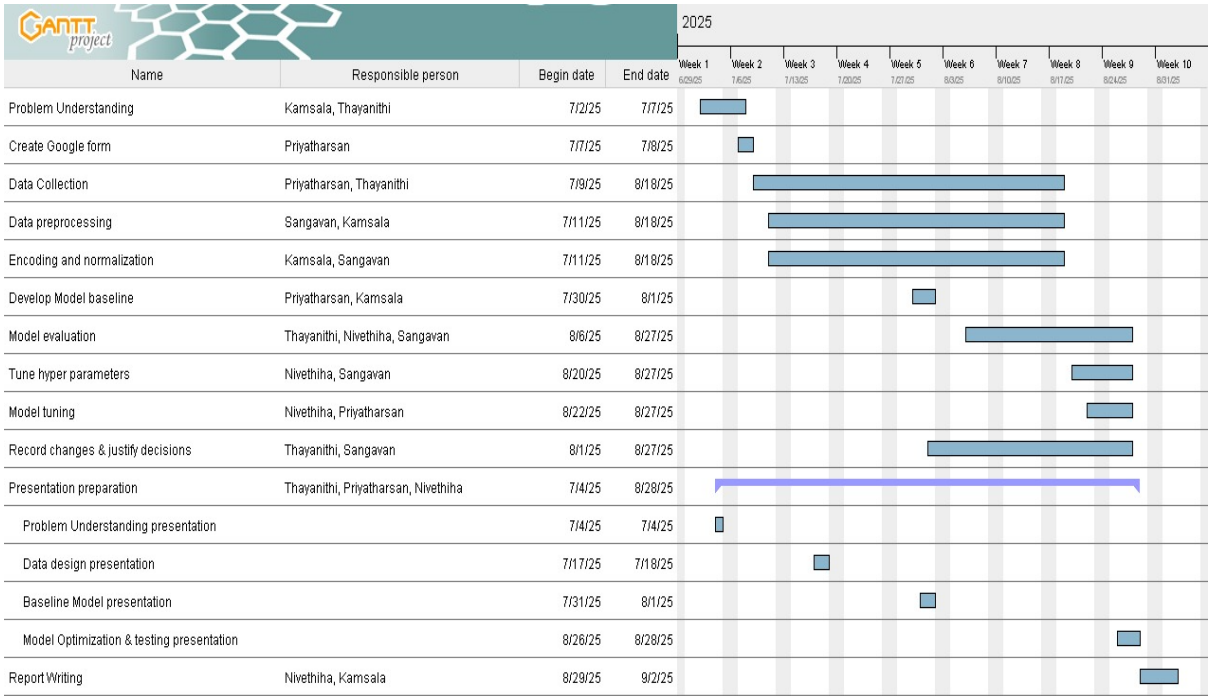
At the same time, we recognized some limitations. The dataset was relatively small and limited in scope. Important factors such as motivation, stress, or family background were not included, even though they strongly influence academic outcomes. Grouping grades into categories (A–E) also reduced detail, as it did not capture smaller differences, such as between an A+ and an A–.

Looking to the future, we believe the system could be improved by expanding the dataset with more features, including personal and psychological factors. More advanced models such as recurrent neural networks (RNNs) or long short-term memory (LSTM) networks could be tested to track changes in study habits over time. Hybrid approaches combining neural networks with other techniques may also improve performance. On a practical level, this model could be developed into a dashboard for teachers and advisors, offering real-time predictions and guidance.

In conclusion, this project highlighted the importance of step-by-step improvement, from building a baseline model to optimizing it through Bayesian and grid search methods. It showed the value of careful data preparation, multiple evaluation metrics, and strategies to reduce overfitting. Most importantly, it demonstrated how predictive models can make a real difference in education by giving early warnings and actionable insights to both teachers and students.

7. Team Work and Contribution

Table:01– Team Contribution



Name	Responsible person	Begin date	End date
Problem Understanding	Kamsala, Thayanithi	7/2/25	7/7/25
Create Google form	Priyatharsan	7/7/25	7/8/25
Data Collection	Priyatharsan, Thayanithi	7/9/25	8/18/25
Data preprocessing	Sangavan, Kamsala	7/11/25	8/18/25
Encoding and normalization	Kamsala, Sangavan	7/11/25	8/18/25
Develop Model baseline	Priyatharsan, Kamsala	7/30/25	8/1/25
Model evaluation	Thayanithi, Nivethiha, Sangavan	8/6/25	8/27/25
Tune hyper parameters	Nivethiha, Sangavan	8/20/25	8/27/25
Model tuning	Nivethiha, Priyatharsan	8/22/25	8/27/25
Record changes & justify decisions	Thayanithi, Sangavan	8/1/25	8/27/25
Presentation preparation	Thayanithi, Priyatharsan, Nivethiha	7/4/25	8/28/25
Problem Understanding presentation		7/4/25	7/4/25
Data design presentation		7/17/25	7/18/25
Baseline Model presentation		7/31/25	8/1/25
Model Optimization & testing presentation		8/26/25	8/28/25
Report Writing	Nivethiha, Kamsala	8/29/25	9/2/25

8. References

[1] Chicho, Bahzad & Sallow, Amira. (2021). A Comprehensive Survey of Deep Learning Models Based on Keras Framework. Journal of Soft Computing and Data Mining. 2.10.30880/jscdm.2021.02.02.005.

[2] Acta Scientific COMPUTER SCIENCE Volume 4 Issue 3 March 2022 Data Analysis Using Pandas Library of Python Rupal Snehkunj1* and Khushboo Vachiyatwala2 https://actascientific.com/ASCS/pdf/ASCS-04-0236.pdf

[3] Leelaluk, Sukrit & Tang, Cheng & Minematsu, Tsubasa & Taniguchi, Yuta & Okubo, Fumiya & Yamashita, Takayoshi & Shimada, Atsushi. (2024). Attention-Based Artificial Neural Network for Student Performance Prediction Based on Learning Activities. IEEE Access. PP. 1-1. 10.1109/ACCESS.2024.3429554.

[4] Ibarra García, Ernesto & Mora, Pablo. (2011). Model Prediction of Academic Performance for First Year Students. 169-174. 10.1109/MICAI.2011.28.

[5] S. Huang and N. Fang, "Work in progress: Early prediction of students' academic performance in an introductory engineering course through different mathematical modeling techniques," 2012 Frontiers in Education Conference Proceedings, Seattle, WA, USA, 2012, pp. 1-2, doi: 10.1109/FIE.2012.6462242.

9. GitHub Link

Cleaned and comment source code:

https://github.com/Nivethiha-Thevarasa/OurFinalModel