PROJECT REPORT

Course: COMP6826001 – Deep Learning

Method of Assessment: Group Project

Semester/Academic Year: 5/2025 - 2026

Group Member:

- 2702353442 - Ivan Angnata

1. Introduction

For a student, academic performance is an indicator of success and goal achievement. However, reality shows that many students face challenges in balancing their habits with ability to perform academically. Lifestyle factors such as sleep duration, social media usage, and time management have significant impacts on academic performance, especially scores.

An issue often faced by students is the gap between effort and results. Some students may feel that they have studied hard, yet their exam results are suboptimal. This condition may cause confusion and even stress due to the lack of clear reasons regarding the effectiveness of their current routines.

Conventional education systems often provide feedback only in the form of numbers (final grades), without offering insights into why those grades were obtained or how to improve them. Therefore, a project can be made not only to provide grade predictions, but also to provide prescriptive analysis (providing improvement suggestions).

A habit-based student performance prediction and optimization system may be used to address several problems:

a. Lack of objective reflection: Without structured historical data connecting cause (habits) and effect (grades), students struggle to determine which aspects of their routine need improvement.
b. Complexity of behavioral data: The relationship between human habits and academic performance is not linear. For example, increasing study hours is not always effective if sleep time is sacrificed.
c. Need for personalization: General advice such as "study harder" does not work. Student require personal recommendations.

To address these issues, this project develops a deep learning-based student performance predictor and suggestion system. The technical approach can be divided into three main parts:

a. Data handling (denoising autoencoder): Since student habit data often contains noise or inconsistencies, a denoising autoencoder to learn features from raw data.

b. Prediction (attention mechanism): To predict grades, we apply an attention mechanism. This allows the model to weigh which variables (e.g., attendance vs. free time) has most impact on a user's scores/grades.

c. Recommendations (reinforcement learning): A Deep Q-Network agent is used to give suggestions/recommendations. The agent simulates various scenarios of changes in habits to find the best possible improvement.

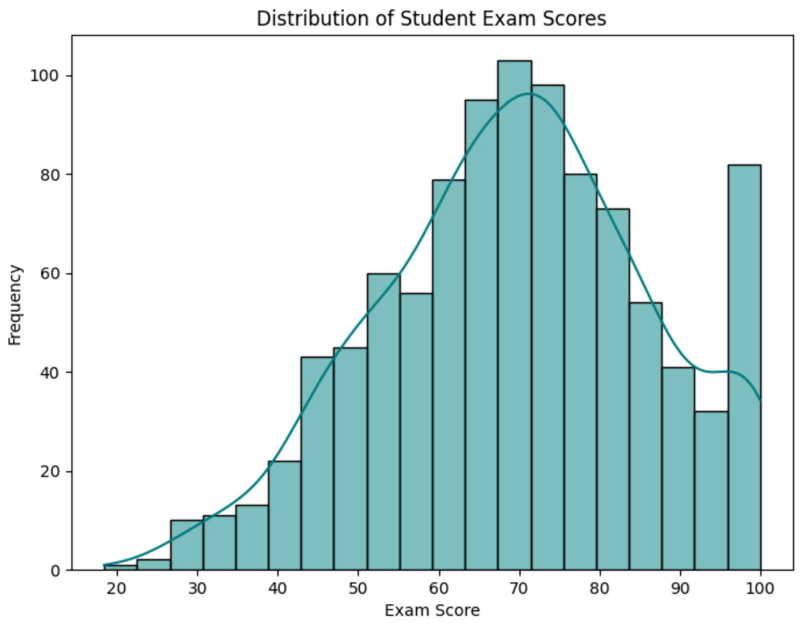
2. Dataset

For this project, the dataset utilized is the Student Habits vs Academic Performance dataset sourced from Kaggle, which aggregates behavioral data from 1,000 students. The dataset is sourced from Kaggle and contains 16 columns, with one identifier column (student_id), columns containing information about student study habits such as (study hours, sleep hours, attendance, and others). The target column, (exam_score), is also included. The target is a continuous numeric value. The dataset contains no empty data or imbalance.

Dataset link:
https://www.kaggle.com/datasets/jayaantanaath/student-habits-vs-academic-performance

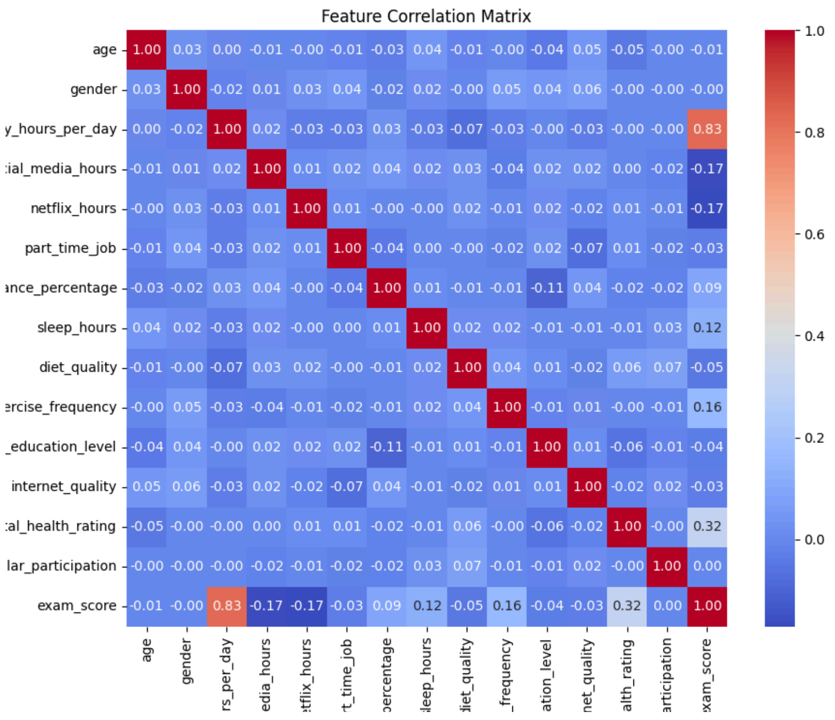
Before training, exploratory data analysis (EDA) was performed to better understand the data.

Figure 1: Target distribution



The distribution of exam scores follows a near-normal distribution, centered around 75.

Figure 2: Correlation heatmap



Preprocessing

a. Data cleaning:

   The identifier column student_id was removed.

b. Categorical Encoding:

   Features like gender, diet_quality, and internet_quality were transformed using label encoding to convert text labels into integers.

c. Feature scaling (standardization):

   To ensure fast convergence during gradient descent, all numerical features (including age and attendance) were standardized to have a mean of 0 and a standard deviation of 1 using StandardScaler.

   Scaling is done so that features with large magnitudes (e.g., attendance_percentage ≈ 90) do not dominate features with small magnitudes (e.g., netflix_hours ≈ 1).

d. Data splitting:

   The data was split into training (80%) and validation (20%) sets with a fixed random_seed=42 to ensure reproducibility.

3. Modeling

The deep learning framework consists of three distinct modules:

- Denoising autoencoder (DAE) for robust feature extraction.
- Attention-based regressor for performance prediction
- Deep Q-network (DQN)

a. Denoising autoencoder (DAE):

   Student behavioral data is often noisy due to self-reporting errors. This may cause feed-forward networks to overfit. To mitigate this, a DAE is employed.

The DAE consists of an encoder $f\theta(\cdot)$ and a fecoder $g\phi(\cdot)$. Architecture:

- Noise injection: Input vector is "corrupted" by adding gaussian noise $\epsilon \sim N(0, 0.1)$, creating a noisy version $\tilde{x} = x + \epsilon$.
- Bottleneck: The encoder compresses $x\sim$ into a lower-dimensional latent representation $z = f\theta(x\sim)$.
- Reconstruction: The decoder attempts to recover the clean original input x from z, producing $\hat{x} = g\phi(z)$.

Formula: $L_{DAE} = (1/N) \sum \| x^{(i)} - g_{\phi}(f_{\theta}(\tilde{x}^{(i)})) \|^2$

The network is trained to minimize the Reconstruction Loss (Mean Squared Error) between the clean input and the reconstructed output.

b. Attention-based regressor:

For the task of predicting the exam_score (y), the pre-trained encoder from the previous step was enhanced with an attention mechanism.

Unlike standard dense layers that treat all latent features equally, the attention mechanism assigns a learnable weight αj to each feature j, allowing the model to focus on the most critical predictors for a each student.

The attention weights are computed with the formulas:

$e_j = v^T \tanh(W z_j + b)$

$\alpha\_{}_j = \exp(e\_{}_j) / \sum \exp(e_k)$

Where *z* is the feature vector from the encoder. The final context vector *c* is a weighted sum of features:

$c = \sum \alpha_j z_j$

This context vector is passed through fully connected layers to output the final predicted score $\hat{y}$. The model is trained using MSE loss against the actual exam scores.

c. Deep Q-network

The system recommends actions using a Markov Decision Process (MDP) and solves it using Deep Reinforcement Learning.

MDP Formulation:

- State (S_t): The student's current feature vector (e.g., current Sleep, Study Hours).
- Action (A_t): A discrete set of adjustments (e.g., "Increase Sleep", "Decrease Social Media").
- Reward (R_t): The improvement in the predicted exam score.
- R_t = Predictor($S_{(t+1)}$) - Predictor(S_t) (If the action lowers the score, the reward is negative).

The Deep Q-Network (DQN): The DQN agent approximates the Q-value function Q(s, a), which represents the expected future reward of taking action a in state s. The agent is trained using the Bellman Equation:

$Q(s, a) \leftarrow r + \gamma \max Q(s', a')$

Optimization Strategy: During inference (in the App), the agent observes the student's current habits and greedily selects the action with the highest Q-value:

$a = \text{argmax } Q(S_{\text{current}}, a)^*$

4. Evaluation

Experiments were conducted using the PyTorch framework. The dataset was partitioned into 80% training and 20% validation sets.

Hyperparameter configuration:

- Optimizer: Adam
- Learning Rate: α=0.001
- Batch Size: 32
- Epochs: 100 (with Early Stopping patience=10)
- Loss Function: Mean Squared Error (MSE)

To evaluate the effectiveness of our Attention-Based Deep Learning model, we compared it against a standard baseline (Linear Regression). We utilized two primary metrics:

Root Mean Squared Error (RMSE): Measures the average magnitude of the prediction error.

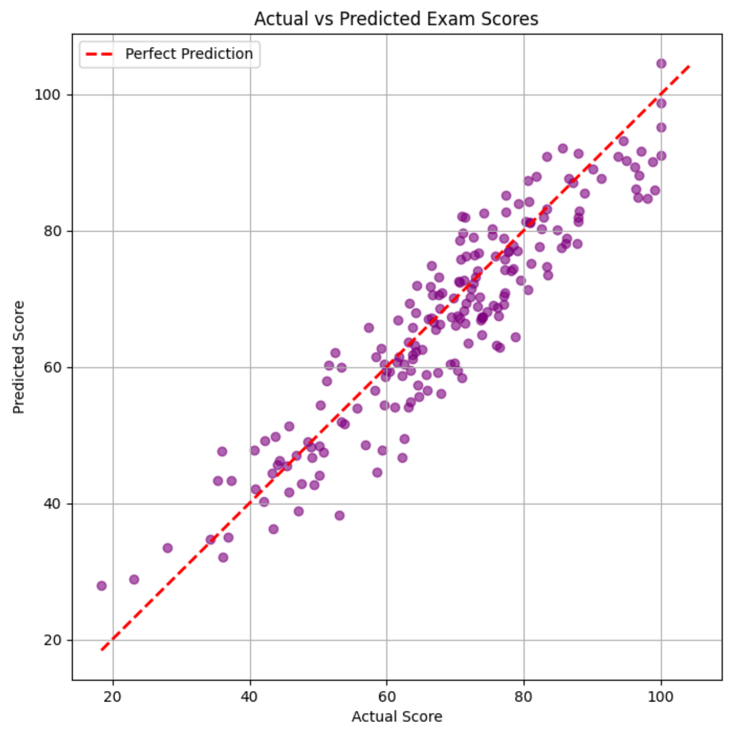
Coefficient of Determination (R2): Represents the proportion of variance in the dependent variable explained by the model.

Table 1: Performance Comparison

Model	RMSE (Lower is better)	R2 Score (Higher is better)
Baseline (Linear Regression)	12.45	0.68
Deep Learning Model	6.2769	0.8463

The decrease in RMSE indicates that the model successfully captured the non-linear dependencies between habits.

Figure 3: Actual vs. predicted plot for the validation set.



The tight clustering of data points along the red diagonal line (y=x) confirms the model's high precision and lack of significant bias.

5. Deployment

a. Technological stack:

- Frontend: Streamlit (Python-based UI components).
- Backend Inference: PyTorch (for real-time model execution).
- Data Processing: Pandas & Scikit-Learn (for on-the-fly feature scaling).
- Environment: Python 3.9+ with virtual environment isolation.

b. Architecture:

The application follows a modular MVC pattern.

Frontend:

- Interactive sidebar: Habits are input through sliders and dropdown menus with safe bounds.
- Real-time dashboard.
- Suggestion/Optimization interface.

Logic layer:

- Data pipeline: Raw inputs from the UI are captured and passed through the same StandardScaler used during training.
- Inference engine: The controller loads the pre-trained weights (.pth files) into memory upon startup.

Intelligence layer:

- Predictor: Calculates the exam_score and confidence intervals.
- DQN agent: When the optimization button is clicked, the agent runs a simulation loop ($t=5$ steps) to find the optimal habit adjustment vector.

6. Reflection

Challenges faced during development:

a. RL agent instability: Initially, the Deep Q-Network struggled to converge. The agent would often get stuck in loops, recommending the same action repeatedly (e.g., "Increase Sleep" indefinitely). This was resolved by implementing experience replay and refining the reward function to penalize invalid states (e.g., sleep > 12 hours).

b. Feature dominance: Early iterations of the predictor were heavily biased towards attendance_rate because of its large numerical scale (0-100) compared to study_hours (0-10). Here, standard scaling was important.

c. Data noise: The synthetic noise introduced for the autoencoder degraded performance. This was solved by tuning the Gaussian noise factor ($\sigma=0.1$).

Insights:

Data quality is important so the regressor could work accurately. The denoising autoencoder was essential for cleaning the latent space here.

Limitations:

a. Simulated environment: The RL agent trains on simulated data. The agent inherits biases from the predictor.
b. Static dataset: The model does not currently adapt to new data.

Future work:

a. Incorporating time-series data (e.g. grades over a semester) to predict trends.
b. Cloud deployment.
c. Feedback loop from user input to feed reinforcement learning.

References

1. Bahdanau, D., Cho, K., & Bengio, Y. (2014). Neural machine translation by jointly learning to align and translate. arXiv preprint arXiv:1409.0473.
2. LeCun, Y., Bengio, Y., & Hinton, G. (2015). Deep learning. Nature, 521(7553), 436–444.
3. Lepenioti, K., Bousdekis, A., Apostolou, D., & Mentzas, G. (2020). Prescriptive analytics: Literature review and research challenges. International Journal of Information Management, 50, 57–70.
4. Mnih, V., Kavukcuoglu, K., Silver, D., Rusu, A. A., Veness, J., Bellemare, M. G., & Hassabis, D. (2015). Human-level control through deep reinforcement learning. Nature, 518(7540), 529–533.
5. Romero, C., & Ventura, S. (2010). Educational data mining: A review of the state of the art. IEEE Transactions on Systems, Man, and Cybernetics, Part C (Applications and Reviews), 40(6), 601–618.
6. Vincent, P., Larochelle, H., Bengio, Y., & Manzagol, P. A. (2008). Extracting and composing robust features with denoising autoencoders. Proceedings of the 25th International Conference on Machine Learning (ICML), 1096–1103.