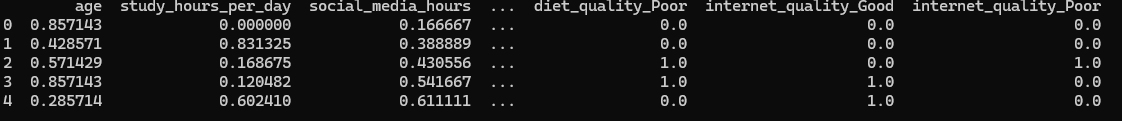
As indicated in the above figure, there are features that are categorial in nature and those that are numeric, including the exam score itself – the value we’re trying to predict. Given that the student\_id is insignificant in what we’re trying to do, we dropped it from the initial dataset we collected.   
Categorical features require to be converted to a mode that can be easily used during the training process. We applied one-hot encoding to these categorical features for that, which includes features such as gender, diet quality, internet quality, etc. Additionally, we standardized (normalized) numerical features (those with continuous values) to ensure that all these values have a similar range.

A screenshot of a computer

AI-generated content may be incorrect.

And standardizing the numerical features and using one-hot encoding, the below is the kind of data we obtain:  
A screen shot of a computer

AI-generated content may be incorrect.  
Important note to make is that, we used the mean-standard deviation normalization in the above figure which affects the predictions negatively, in that, the predictions result in a high MAE value compared to MinMax normalization which results in a relatively lower MAE value which produced the table of features as follows:



**Data Split**:

* 60% training data
* 20% validation data
* 20% test data

4. We implemented the Multilayer Perceptron. This is because it model nonlinear relationships among features in a seamless manner. The objective was to build a sufficiently complex model that could generalize well without overfitting.  
A computer screen with text

AI-generated content may be incorrect.

* The input layer accepts all processed features (numerical + one-hot encoded categorical).
* Two hidden layers (128 and 64 neurons) with **ReLU** activation functions.
* Dropout layers (0.4 and 0.3) added for regularization and to prevent overfitting.
* Output layer contains a single neuron with **linear activation**, suitable for regression.

**Optimizer**: We used the **Adam optimizer**, a widely used gradient descent method that adapts the learning rate during training for faster convergence.

**Loss Function**: Instead of mean squared error (MSE), we used **Huber loss**, which behaves like MSE for small errors and like MAE for large errors, offering robustness to outliers.

**Metric**: We tracked **Mean Absolute Error (MAE)** during training and evaluation, which is more interpretable in the context of exam scores (e.g., "the model is off by 4.1 marks on average").

5.

We tried using the Min-Max normalization in attempt to standardize our numerical features and we got a high error value versus when we used mean standard deviation normalization.  
Figure 5.1 shows the results we got when running min-max and figure 5.2 shows our results when running Mean standard deviation.  
  
  
We then explored with the use of different layers for our MLP and what we got for 3 layers was not giving any lower error values as seen in figure 5.3.1.

We found that 2 hidden layers perform better than 3 hidden layers. For that, we then explored the use of different combinations of different activation functions to see the combinations that give us minimal error values. Figure 5.3.1.2 is the case where we used relu and relu activation function for