Name: Rishabh Gupta
Roll Number: DA25M024

1. Data Engineering

Dataset Overview

The dataset had 5,000 training text responses with scores from 1 to 10, and 3,638 test texts where score prediction was needed. Every row had four main things: a metric name, the user prompt, the model response, and the score. There were more than 50 metrics in total. But the main issue was that the data was not balanced at all.

Original Distribution Problem

Most of the samples were in score 9 and 10, almost 91%, so the distribution was very right–skewed. If I trained directly on that, the model would mostly learn to predict only high scores, which is useless.

Score	Samples	Percentage
1	5	0.10%
2	8	0.16%
3	7	0.14%
4	3	0.06%
5	1	0.02%
6	45	0.90%
7	95	1.90%
8	259	5.18%
9	3,124	62.48%
10	1,453	29.06%

2. Data Augmentation

Goal

The target was to increase data size from 5k to around 50k and also fix the class imbalance. I wanted a more normal-type distribution with proper amount of low, medium and high score samples.

Augmentation Work

I used a mix of undersampling, oversampling and relabeling. Big classes like 9 and 10 were reduced. Very small classes were increased by sampling with replacement. For score 5 (only one sample), I borrowed nearby samples and relabeled them.

Metric Mismatch Method

One big issue was low-score examples. So I made a new method: metric mismatch. Basically, if a high-scoring response was meant for “grammar”, I paired it with some wrong metric like “technical accuracy”. Because the response doesn’t match that metric, it becomes a low-quality example logically. I created around 8 such mismatch versions per high score sample and gave them scores like 0, 1, or 2. This made the model actually learn what "metric alignment" means.

Embedding Handling

Even though dataset became 50k rows, I reused same 5k embeddings using a lookup map. So duplicates or mismatch samples point to the right embedding index without re-computing anything.

Final Distribution

Score	Samples	Percentage
1	1,470	2.94%
2	2,005	4.01%
3	3,355	6.71%
4	5,265	10.53%
5	6,970	13.94%
6	5,390	10.78%
7	7,985	15.97%
8	11,435	22.87%
9	5,595	11.19%
10	535	1.07%
Total	**50,005**	**100%**

The whole distribution became more stable and kind of bell-shaped. The mean dropped from 9.2 to around 6.1.

3. Sampling Strategy

I used stratified sampling so that all score classes remain in same proportion in both training and validation set. Final split was 95% training and 5% validation.

4. Text Embeddings (Gemma)

I used Google Gemma embedding model. Each metric and each response becomes a 384-d vector, then I normalized both metric and text embeddings separately. These embeddings are what the model actually sees.

5. Model Architecture

The model input is 768 dimensions (metric + text embeddings). I used a feed-forward neural network with hidden layers of size 1024, 768, 512 and 256. ReLU activation was used, and batch norm + dropout helped with regularization. Softmax was used for the final 10 classes. Loss function was cross-entropy.

6. Training Details

Training ran for 30 epochs using Adam optimizer. Learning rate started at 0.0001 and was reduced using cosine annealing. Batch size was 256.