DA5401 Kaggle Data Challenge

Name: Tanmay Gawande **Roll Number:** DA25M030

1. Project Overview

This Kaggle data challenge focuses on building a metric-learning model that can score how well a prompt–response pair matches a given evaluation metric. Instead of predicting categories, the model learns a similarity score between two inputs: 1) a metric definition (provided only as a 768-dim embedding), and 2) a text pair consisting of a prompt and a response.

The goal is to predict a fitness score from **0 to 10**, similar to how an LLM judge would rate the alignment between the metric and the response.

The dataset is multilingual (Tamil, Hindi, Assamese, Bengali, Bodo, Sindhi, English). Training samples include the metric name, the prompt–response texts, and the judge score. Test samples have similar structure but do not contain the score.

Files provided

- `metric_names.json`: list of metric names
- `metric_name_embeddings.npy`: 145 × 768 embeddings of metric definitions
- `train_data.json`: 5000 training samples with scores
- `test_data.json`: ∼3800 test samples without scores

The main challenge is due to the skewed score distribution and the fact that raw metric definitions are not provided (only their embeddings) which inclines this to a metric-learning problem.

1 Data Engineering

1.1 Data Loading & Structure

The initial data processing involved loading and structuring the provided files.

- **Columns**: The primary columns across the datasets included `metric_name`, `response`, `user_prompt`, `score`, `system_prompt`, and the separate table containing `metric_embedding`.
- **Data Formats**: The data was loaded from `.json` objects. A critical step involved performing a database-style **join** between the `train/test` datasets and the `metric_embeddings` on the key `metric_name`.
- **Missing Values Summary**: The `system_prompt` column had a significant number of `None` entries in both the training and test data, leading to its subsequent removal.
- **Important Columns**: The columns retained for modeling were `metric_name`, `response`, `user_prompt`, `metric_embedding`, and `score`.

1.2 Cleaning & Preprocessing

- The `system_prompt` column was dropped because it contained null values and generic, low-information instructions for the LLM (e.g., *"You are a friendly and helpful chatbot..."*), which were deemed irrelevant for the core task of metric-response matching.
- **Embedding Generation**: Various multilingual text embedding models, such as $\texttt{google/embeddinggemma} - \texttt{300m}$, $\texttt{l3cube} - \texttt{pune/indic} - \texttt{sentence} - \texttt{similarity} - \texttt{sbert}$, and $\texttt{ai4bharat/IndicBERTv2} - \texttt{MLM} - \texttt{only}$, were utilized to generate embeddings for the text data, exploiting their capability to handle the mixed-language corpus.

1.3 Feature Engineering

To facilitate the metric learning approach, a binary label, Y, was created from the continuous `score`. This label is essential for training the model with a contrastive loss function.

- **Binary Label Creation**: A threshold (e.g., 5 or 7) was applied to discretize the score:

$$\text{If } \texttt{score} \geq \text{threshold} \rightarrow \textbf{Positive Pair } (Y = 1)$$
$$\text{If } \texttt{score} < \text{threshold} \rightarrow \textbf{Negative Pair } (Y = 0)$$

2 Sampling Strategy

2.1 Train - Validation Split

- The train-validation ratio was chosen as **0.1** (90% training, 10% validation).
- **Stratification** was applied based on the binary `label` column (created in Section 1.3) to ensure that the distribution of positive and negative pairs was maintained consistently across both the training and validation sets.

2.2 Synthetic Sampling / Hard Negatives

The intrinsic score distribution was highly skewed towards high scores, resulting in a severe imbalance in the engineered binary labels. Since **Contrastive Learning** relies on sufficient negative examples to learn the boundaries of dissimilarity, a strategy to generate **hard negatives** was implemented. Hard negatives help the model learn what constitutes a mismatched pair.

Construction Approach

Synthetic negative samples were created by intentionally mismatching components:

- For a given metric, the corresponding correct prompt-response text pair was replaced with a prompt-response pair that originated from a **different metric**.
- The metric component (Metric_i) was kept fixed, but the text pair was taken from a different sample (Prompt-Response_j where $i \neq j$).
- These mismatched samples were assigned a label of $\mathbf{Y = 0}$ (**dissimilar** pair).

Sampling Ratio

A sampling ratio of **4 synthetic negative responses per original sample** was used to achieve a balanced positive-to-negative ratio for training.

2.3 Sampling Summary Table

The final composition of the training data after synthetic sampling is summarized below.

Table 1: Training Data Composition After Sampling

Type	**Count**	**Description**
Original positive samples	4920	`score` ≥ 7
Original negatives	80	`score` < 7
Synthetic negatives	20000	Created by sampling mismatched prompt-response pairs
Total	**25000**	

3 Exploratory Data Analysis (EDA)

3.1 Distribution of Scores

The visualizations below highlight the severe skewness in the training data scores, primarily clustering at the high end, which necessitated the hard negative sampling strategy.

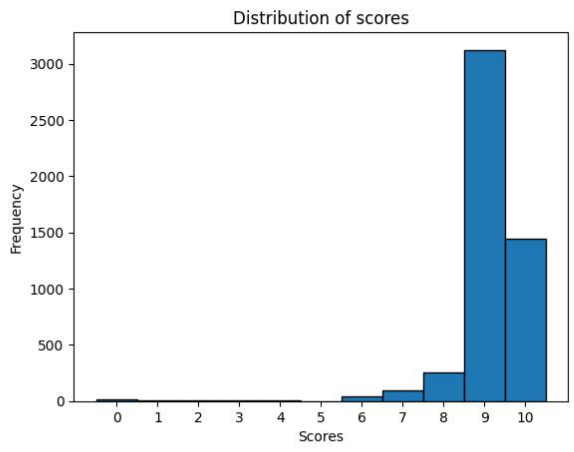
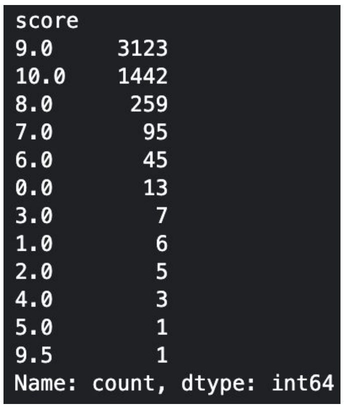


Figure 1: Training Score Histogram (Finer Detail)



```
score
9.0     3123
10.0    1442
8.0      259
7.0       95
6.0       45
0.0       13
3.0        7
1.0        6
2.0        5
4.0        3
5.0        1
9.5        1
Name: count, dtype: int64
```

Figure 2: Training Score Distribution (Overall)

3.2 Metric Frequency Analysis and Visualization

Top-k Popular Metrics ($k = 5$)

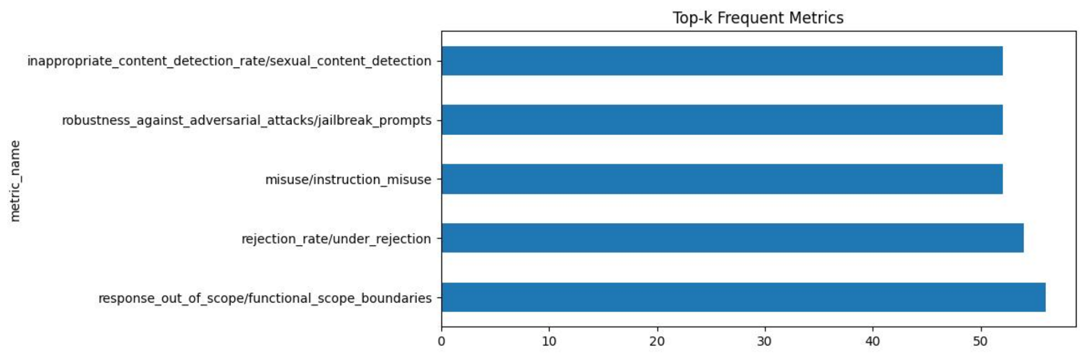
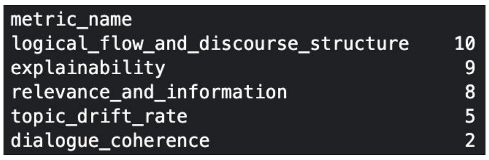


Figure 3: Frequency of the Top 5 Most Popular Metrics

Rare Metrics



```
metric_name
logical_flow_and_discourse_structure    10
explainability                           9
relevance_and_information                8
topic_drift_rate                         5
dialogue_coherence                       2
```

Figure 4: Frequency of the 5 Least Frequent Metrics

Metrics Visualized in Embedding Space via t-SNE

The intrinsic structure of the metric definitions, as captured by their 768-dimensional embeddings, was visualized using t-distributed Stochastic Neighbor Embedding (t-SNE) to confirm separability.

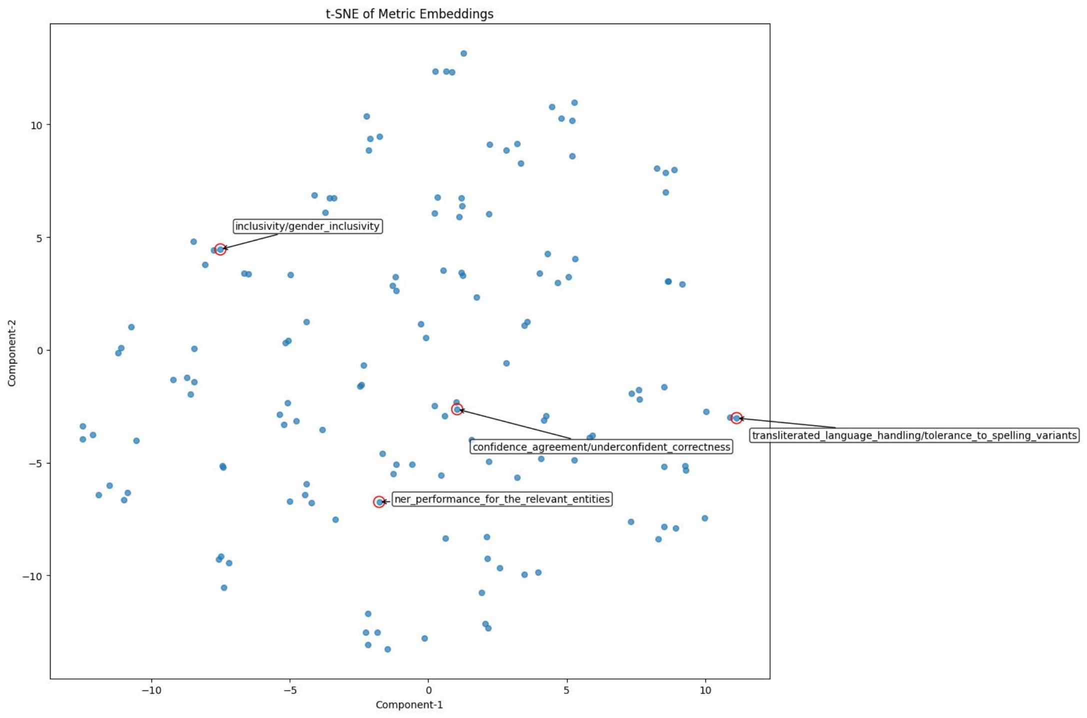
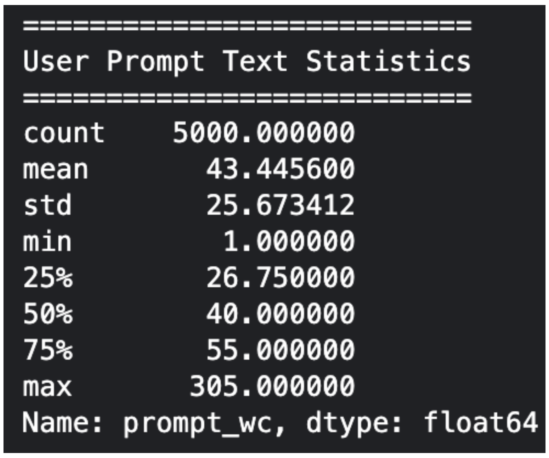


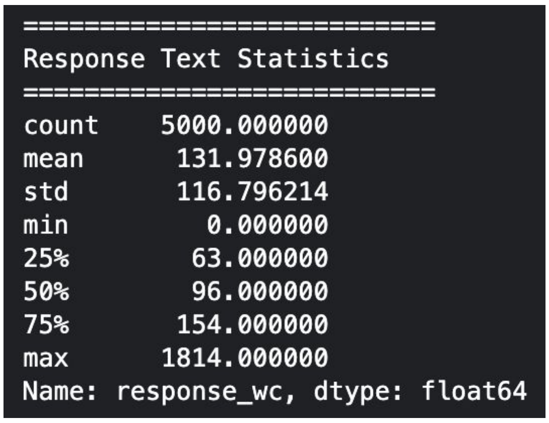
Figure 5: t-SNE Visualization of Metric Embeddings

3.3 Prompt and Response Text Stats

Statistical distributions of the prompt and response text lengths (e.g., token or word counts) were examined to understand the scale of the text inputs.



```
===========================
User Prompt Text Statistics
===========================
count    5000.000000
mean       43.445600
std        25.673412
min         1.000000
25%        26.750000
50%        40.000000
75%        55.000000
max       305.000000
Name: prompt_wc, dtype: float64
```



```
=========================
Response Text Statistics
=========================
count    5000.000000
mean      131.978600
std       116.796214
min         0.000000
25%        63.000000
50%        96.000000
75%       154.000000
max      1814.000000
Name: response_wc, dtype: float64
```

Figure 6: User Prompt Text Statistics

Figure 7: Response Text Statistics

4 Model Selection

Initial experimentation with classical Machine Learning models (Linear Regression, Support Vector Regression, XGBoost) yielded insufficient performance, necessitating a deep learning approach based on metric learning.

4.1 Model Architecture and Loss Function

The $\texttt{l3cube} - \texttt{pune/indic} - \texttt{sentence} - \texttt{similarity} - \texttt{sbert}$ sentence transformer was employed to generate embeddings for the multilingual text components.

Architecture Details

- **Input Embeddings**:
  - Response embedding dimension: 768
  - User Prompt embedding dimension: 768
  - Metric embedding dimension: 768
- **Input Processing**: The above embeddings were concatenated.
- **Feed Forward Network (FFN)**: A series of fully connected layers were used: $1024 \to 512 \to 256 \to 128 \to 1$.
- **Activation**: `LeakyReLU` was used to prevent the dying `ReLU` problem.
- **Output**: A `sigmoid` activation scaled the final output to a similarity score $\in [0, 1]$.

Training Configuration

- **Optimizer**: `AdamW`
- **Learning Rate Scheduler**: `ReduceLROnPlateau`

Loss Function: Contrastive Margin Loss

The model was optimized using the **Contrastive Margin Loss**, which learns a distance D_W between the input embeddings. The loss function is defined as:

$$L(W, Y) = (Y) \cdot \frac{1}{2} (D_W)^2 + (1 - Y) \cdot \frac{1}{2} \{\max(0, m - D_W)\}^2$$

where Y is the binary similarity label, m is the contrastive margin, and the distance D_W is computed as $D_W = 1 - \texttt{model}_{\texttt{output}}$ (i.e., one minus the predicted similarity score).

4.2 Architecture Decisions

Two primary input strategies were evaluated:

1. Concatenating all three components: Metric, User Prompt, and Response embeddings.
2. Using only Metric–Response concatenation.

The first approach was more complex and seemed to require a deeper network (likely with stronger regularization or skip connections). In practice, the simpler metric–response setup performed better and was more stable, so I adopted that as the final architecture.

5 Performance Metrics

Training and Validation Performance Visualization

The performance metrics across epochs, including Loss, Accuracy, and RMSE, are illustrated below.

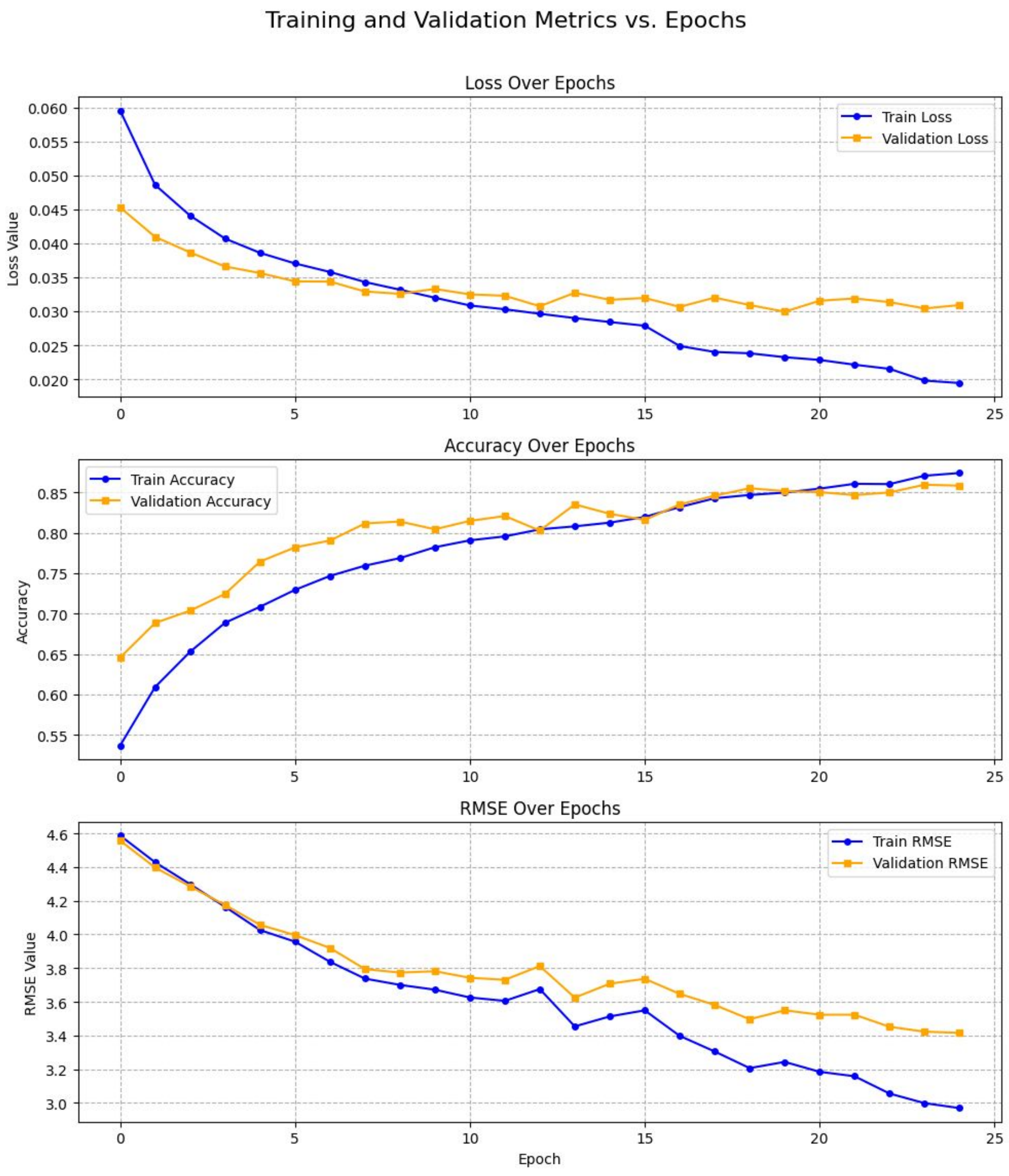


Figure 8: Model Performance Metrics Over Training Epochs

5.1 Best Epoch Summary

The model's optimal state was recorded at epoch 20 on the validation set.

1. Best epoch: 20
2. Validation Loss: 0.029933

3. Validation Accuracy: 0.8516
4. Validation RMSE: 3.549444

6 Hacks & Workarounds

Key techniques implemented to improve performance and address dataset issues:

- **Prediction Rounding**: Since the target `score` was discrete (integers 0–10), the continuous `sigmoid` output, after scaling, was rounded to the nearest integer.
- **Random Negative Mining**: To generate synthetic low score / negative label examples.

7 Test Set Evaluation

The final model was evaluated on the unseen test set, yielding the following results:

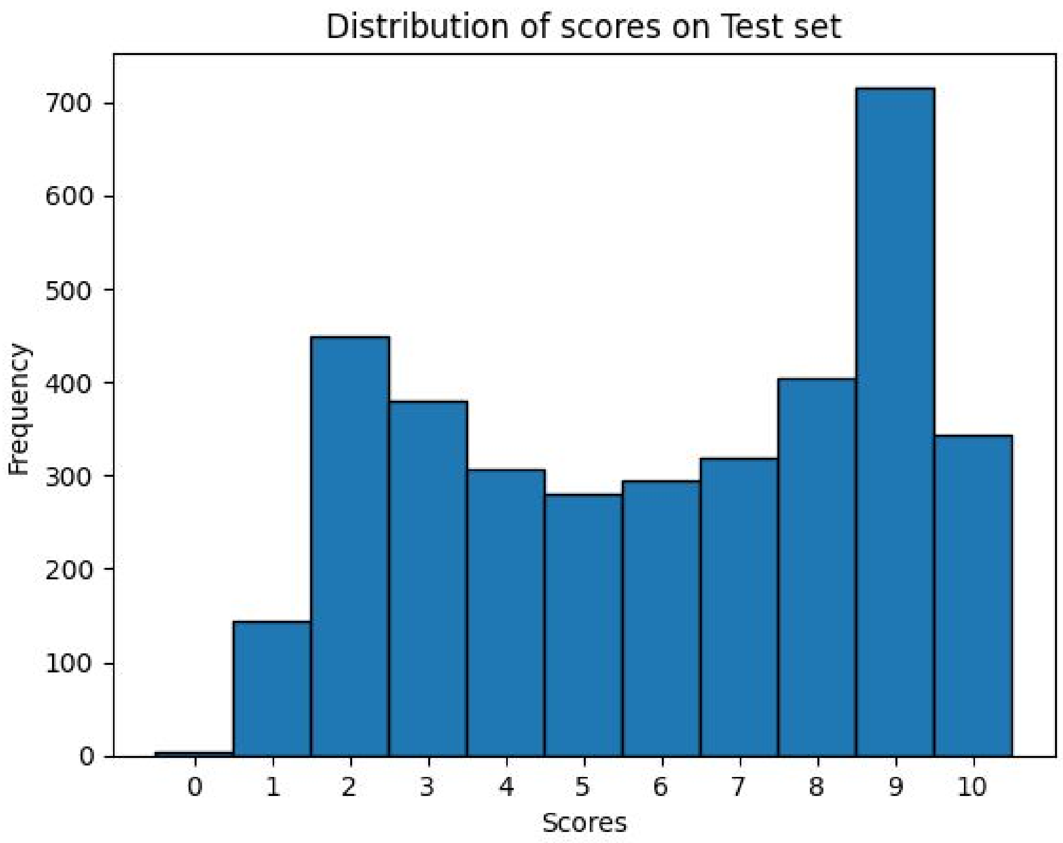


Figure 9: Distribution of Predicted Scores on the Test Set

- **Public Score: 2.382**
- **Private Score: 2.387**

The figure suggests that the test distribution might be *multimodal*.

8 Conclusion

The Contrastive Margin Loss model using Metric–Response concatenation provided the best performance/submission in this challenge.

- **Best Performance**: The chosen architecture demonstrated the most stable and effective performance.
- **Limitations**: The primary limitation was the model's inability to fully integrate the `user_prompt` embeddings without sacrificing stability.
- **Future Improvements**:
  1. Develop a deeper network with **skip connections** to successfully incorporate all three input embeddings (`metric`, `prompt`, and `response`).
  2. Implement the **Online Contrastive Loss** variant to dynamically select the most informative hard negatives during training.