INDIAN INSTITUTE OF TECHNOLOGY MADRAS

End-Semester Project Report

DA5401

Mohmad Yaqoob

Roll No: DA25M017

M.Tech Data Science and Artificial Intelligence

Contents

1. Introduction

Evaluating the quality of conversational AI systems requires checking whether a model's response aligns with the intent of a specific evaluation metric, such as coherence, correctness, safety, or relevance. In real evaluation pipelines, this is done by an LLM judge that compares a Prompt–Response pair with a written Metric Definition and assigns a fitness score between 0 and 10. While this approach is accurate, it is slow, costly, and difficult to scale when thousands of multilingual examples must be evaluated.

This Data Challenge focuses on replacing this expensive LLM-based judge with a machine learning model. The task is to predict the fitness score using only:

- the embedding of the metric definition, and
- the embedding of the prompt–response text pair.

Since the original metric definition text is not provided—only its embedding—this becomes a metric learning problem where the model must infer semantic similarity directly from dense vectors. The dataset spans multiple languages (Tamil, Hindi, Assamese, Bengali, Bodo, Sindhi, and English), making the problem more challenging.

The goal of the project is to build a regression model that learns the relationship between the metric embedding and the prompt–response embedding, and predicts the fitness score in the range 0–10 with low RMSE. To address this, the project includes:

- generating text embeddings for prompts, responses, and system prompts using Sentence-Transformer models,
- constructing interaction features to capture metric–text relationships,
- performing EDA to understand score imbalance and embedding behaviour,
- designing a deep ResNet-MLP model suitable for high-dimensional regression,
- using training improvements such as SWA, EMA, and KL-based distribution loss,
- applying post-processing methods like calibration and quantile mapping.

This combination of embedding engineering, model design, and distribution-aware post-processing enables the system to approximate LLM-judge behaviour efficiently and produce reliable fitness predictions for multilingual conversational AI evaluation.

2. Data Engineering

The dataset provided for the challenge consists of multiple components that must be processed and aligned before model development. Each training record contains a metric name along with a corresponding prompt, system prompt, and model response. Additionally, the dataset includes

pre-computed text embeddings of all metric definitions. Since the raw definitions are not shared, these embeddings represent the only way to understand the intent behind each evaluation metric.

2.1 Data Components

The main data sources used in this project are:

- **metric_name_embeddings.npy**: a (145×768) matrix containing dense embeddings of all evaluation metric definitions.
- **train_data.json**: JSON records containing the metric name, user prompt, system prompt, response text, and the ground-truth fitness score in the range 0–10.
- **test_data.json**: similar to the training data, but without the target score.
- **metric_names.json**: list of unique metric names used to index embeddings.

The text in the prompt, response, and system prompt fields is multilingual (Tamil, Hindi, Assamese, Bengali, Bodo, Sindhi, and English), requiring a language-agnostic embedding model.

2.2 Embedding Generation

To convert textual information into numeric form, we used the `google/embeddinggemma-300m` SentenceTransformer model. For each training and test example, three separate embeddings were generated:

- embedding of the user prompt,
- embedding of the system prompt,
- embedding of the model response.

Since all three contribute to the final meaning of the test instance, they were combined into a single unified vector. The simplest and most stable approach was adopted:

$$\text{TextEmbedding} = \frac{E_{\text{prompt}} + E_{\text{system}} + E_{\text{response}}}{3}$$

This averaged embedding captures the overall semantic content of the entire interaction.

2.3 Aligning Embeddings with Metrics

Each training record specifies a `metric_name`, which is mapped to the corresponding row index in the metric embedding matrix. This allows each sample to be paired with the correct (768-dimensional) metric definition embedding.

Thus, every data point is represented by:

- a metric embedding (768-dimensional),

- a unified text embedding (768-dimensional),
- the target fitness score (training only).

2.4 Preparing Model-Ready Features

After generating embeddings, additional numerical interaction features (such as elementwise differences, products, and cosine similarity) were constructed in the next stage. These steps ensure that both the metric intent and the textual behaviour are numerically represented in a form suitable for the downstream regression model.

Overall, the data engineering pipeline ensures that all textual, multilingual, and semantic information is transformed into a consistent, aligned, and numerically expressive representation before model training.

3. Feature Engineering

To model the semantic relationship between the Metric Definition and the Prompt–Response pair, both embeddings must be transformed into expressive numerical features that capture alignment, similarity, and directional differences. Let M denote the metric embedding and T denote the unified text embedding, each of dimension 768.

3.1 Interaction Features

Several interaction-based features were constructed to help the model learn how the two embeddings relate:

- **Concatenation:** The simplest representation, formed by placing the two embeddings side by side:
$$[M, T]$$
- **Absolute Difference:** Captures how far the text behaviour deviates from the metric intent:
$$|M - T|$$
- **Element-wise Product:** Highlights dimensions where both embeddings exhibit similar semantics:
$$M \cdot T$$
- **Cosine Similarity:** A scalar value indicating global similarity between metric and text embeddings:
$$\cos(M, T) = \frac{M \cdot T}{\|M\| \, \|T\|}$$

3.2 Final Feature Vector

After computing these components, all features were concatenated into a single vector:

$$\text{FinalFeatures} = [M, T, |M - T|, M \cdot T, \cos(M, T)]$$

This produces a dense feature representation of approximately 3073 dimensions:

$$768 + 768 + 768 + 768 + 1 = 3073$$

3.3 Motivation

These engineered features allow the model to capture:

- direct semantic content (via concatenation),
- degree of mismatch (via absolute difference),
- agreement on semantic directions (via product),
- overall alignment score (via cosine similarity).

This richer feature space enables the downstream neural model to learn fine-grained relationships between metric definitions and prompt–response behaviour, which is essential for accurate fitness score prediction.

4. Exploratory Data Analysis (EDA)

A detailed exploratory analysis was performed on the provided embeddings and textual data to understand the structure of the dataset, the behaviour of metric embeddings, and differences between the training and test distributions. Only the essential and meaningful plots are included here.

4.1 Metric Embedding Analysis

The 145 evaluation metric definitions were provided only as embeddings. To understand their structure, several projections and similarity analyses were performed.

4.1.1 PCA Projection of Metric Embeddings

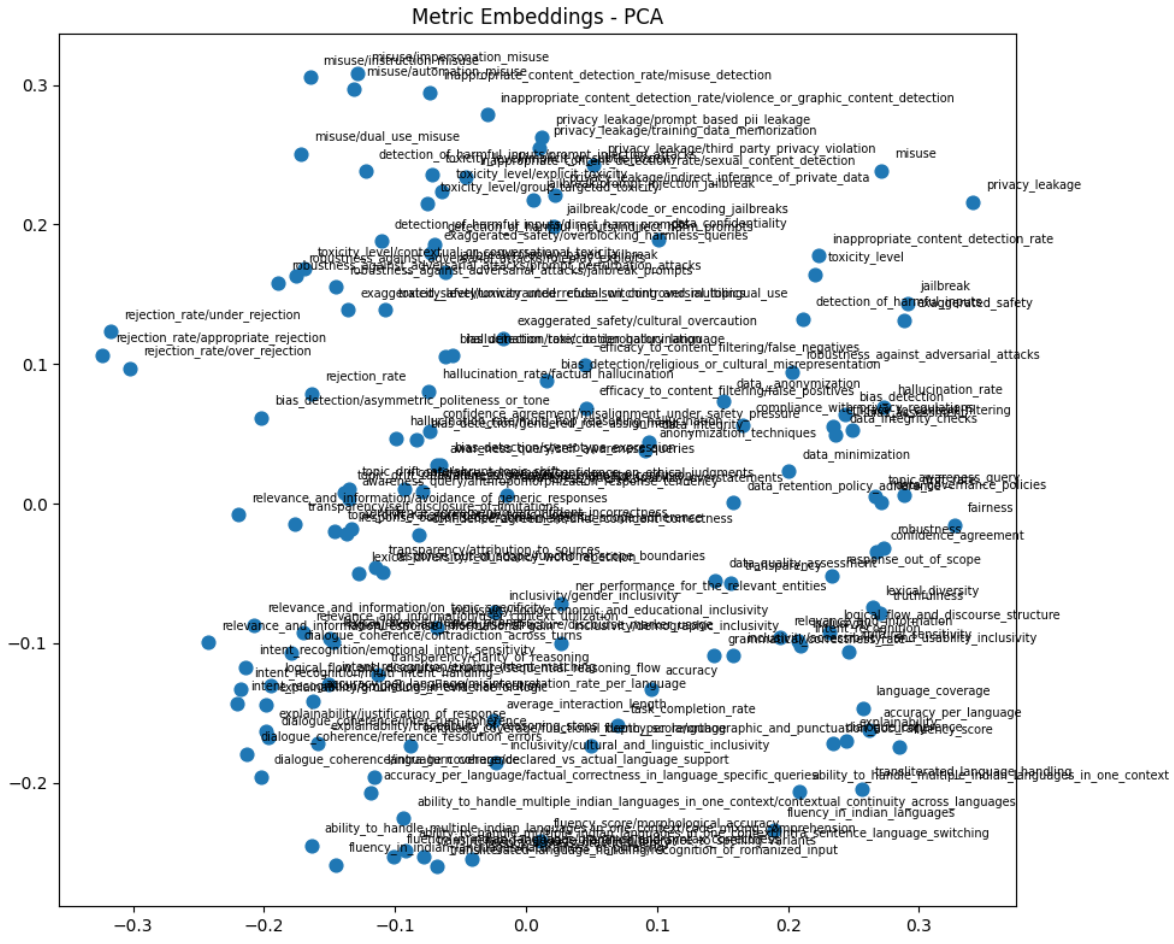

Figure 1: PCA projection of metric definition embeddings. Distinct clusters indicate semantic grouping among metrics.

4.2 Training Data Analysis

4.2.1 Score Distribution

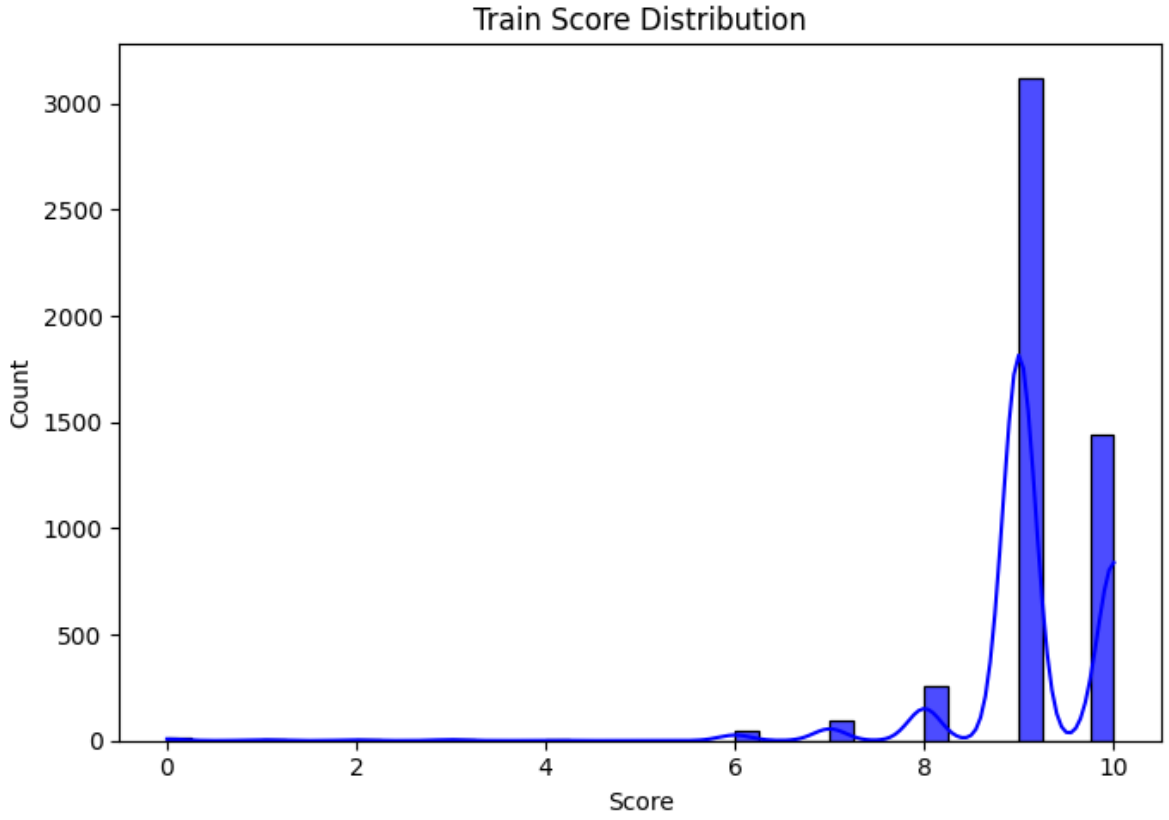

Figure 2: Training score distribution. The scores are highly skewed toward the higher end, consistent with LLM-judge behaviour.

4.2.2 Score Boxplot

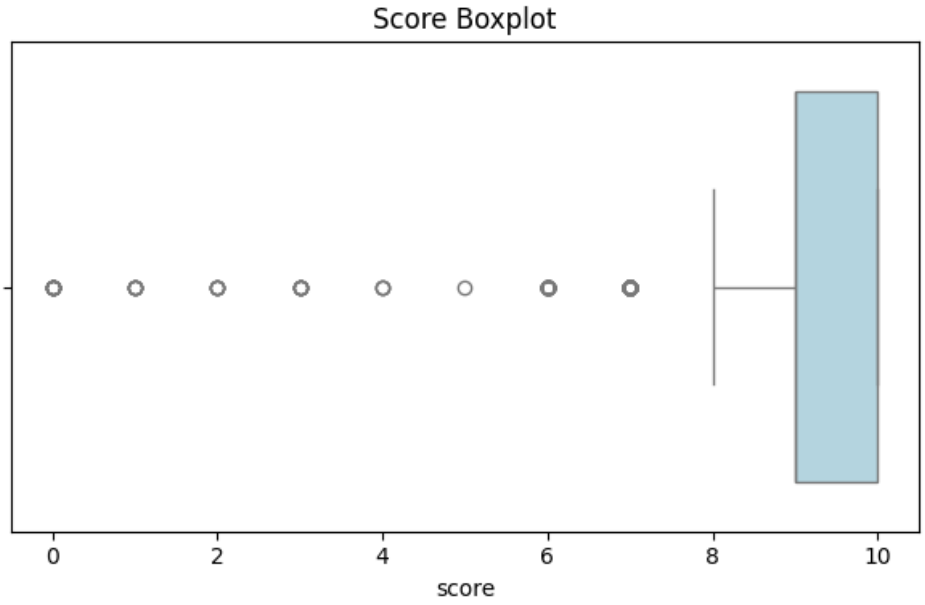

Figure 3: Boxplot of training scores showing concentration near the upper range.

4.2.3 User Prompt Length Distribution

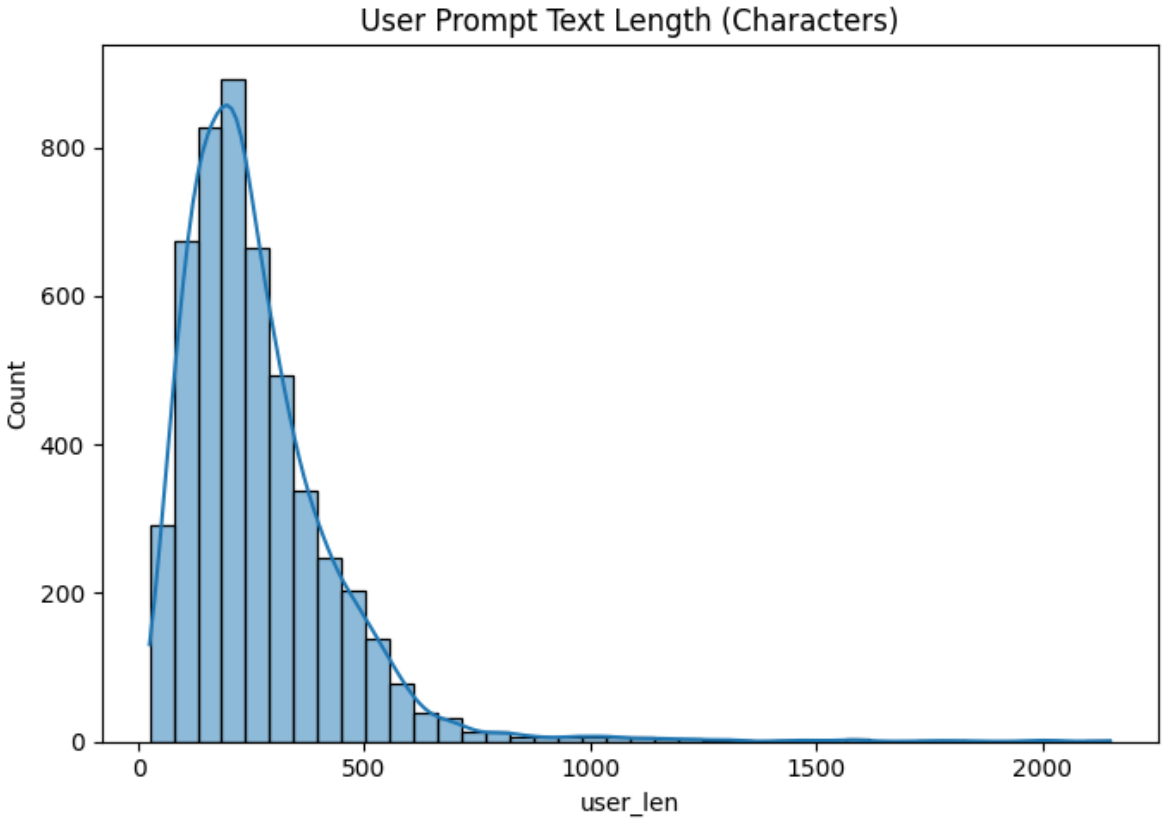

Figure 4: Distribution of user prompt lengths in the training data.

4.2.4 Response Length Distribution

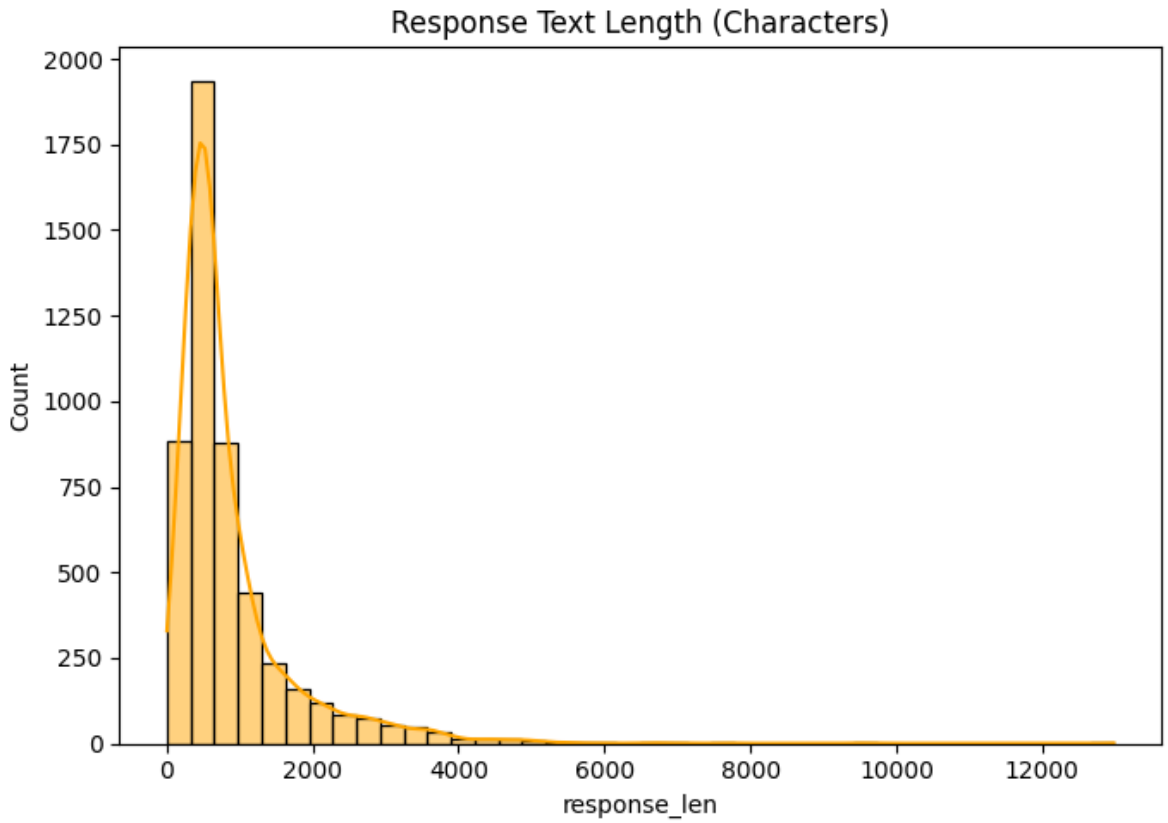

Figure 5: Distribution of model response lengths in the training data.

4.2.5 Metric Name Distribution

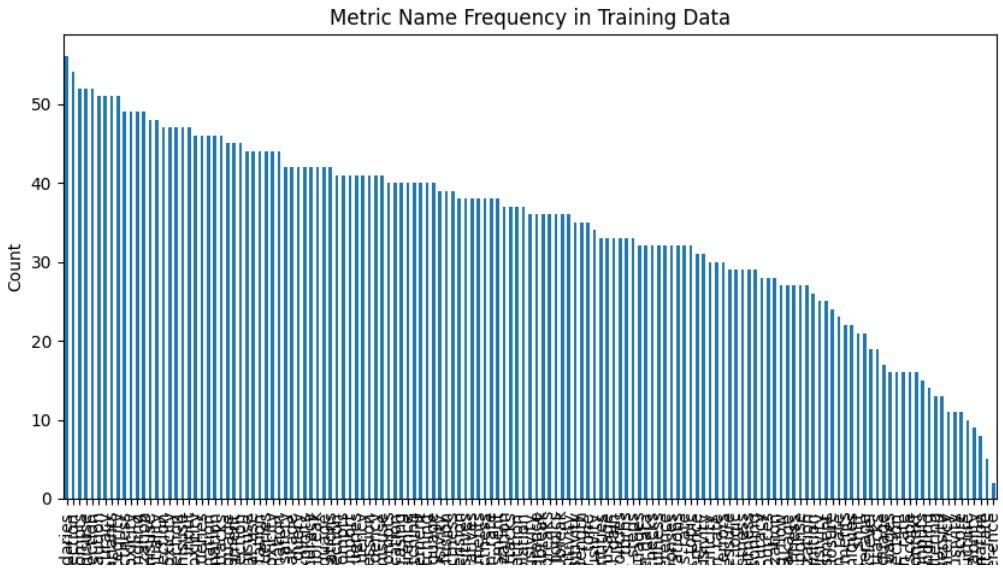

Figure 6: Frequency distribution of metrics in the training set. Some metrics appear significantly more often.

4.3 Train–Test Distribution Comparison

4.3.1 Metric Distribution Comparison

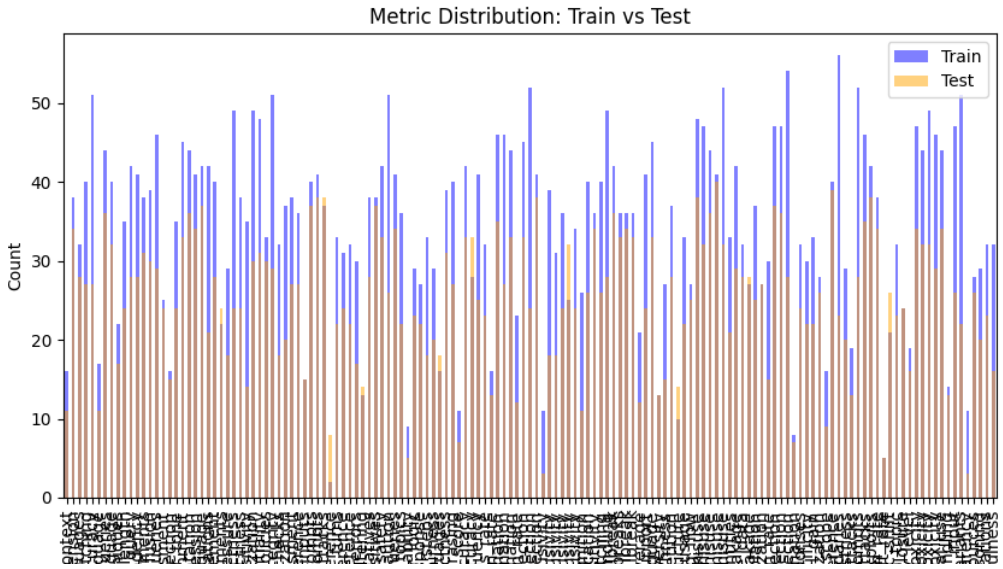

Figure 7: Comparison of metric frequency between training and test sets. Overall distributions are similar.

4.3.2 User Prompt Length Comparison

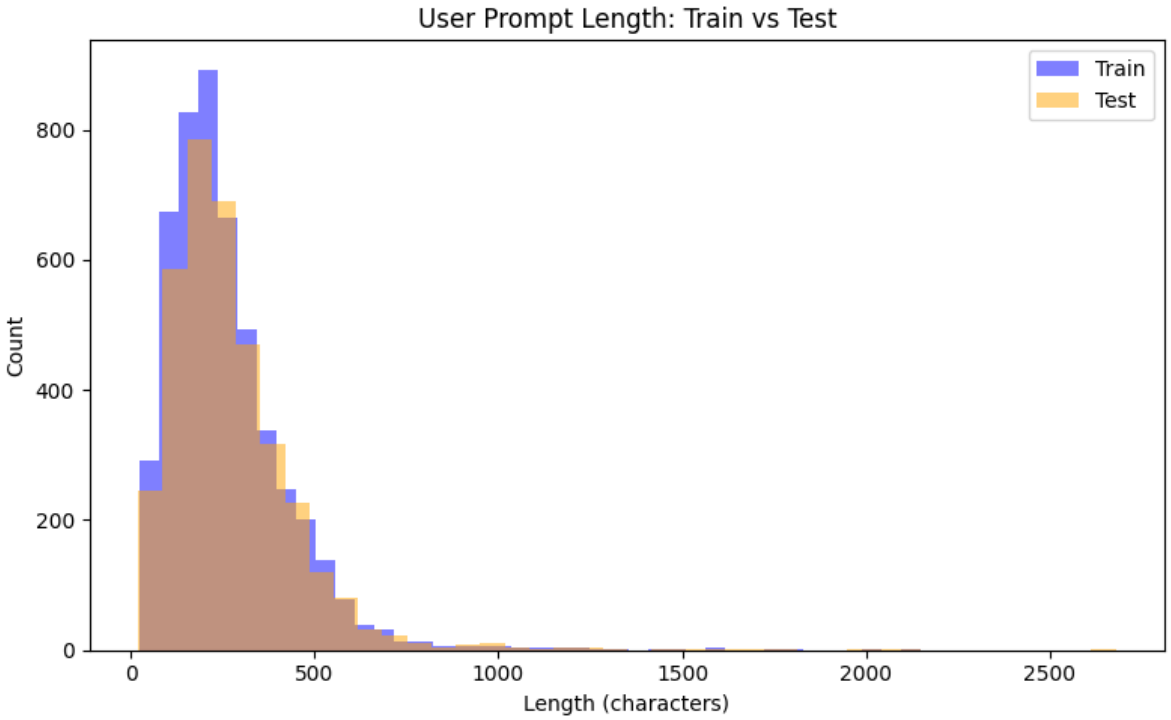

Figure 8: Comparison of prompt length distributions between training and test sets. Both sets show similar text-length behaviour.

This EDA confirmed that the embedding space is structured, the score distribution is skewed, and the train–test distributions are largely aligned—important properties that guided the later modeling decisions.

5. Model Selection

Given the high dimensionality of the embedding space (3073 features per sample) and the nonlinear relationships between metric definitions and prompt–response semantics, a deep neural network was chosen as the primary model. After experimentation with simpler baselines, a custom **ResNet-MLP** architecture was selected due to its stability, expressive power, and strong empirical performance on dense vector regression tasks.

5.1 Motivation

Traditional shallow models such as linear regression or simple MLPs were unable to capture the rich interaction patterns between the metric embeddings and text embeddings. Transformer-based models were unnecessary because the input is already a single dense vector rather than a sequence. A residual MLP offered the right balance: it is computationally efficient, avoids vanishing gradients, and handles large feature spaces effectively.

5.2 Architecture Overview

The final model consists of the following components:

- **Input Projection Layer:** A linear layer followed by LayerNorm, GELU activation, and dropout. This maps the 3073-dimensional input into a 1024-dimensional latent space.
- **Residual MLP Blocks (8 Blocks):** Each block contains:
  - LayerNorm
  - Linear layer expanding to $4\times$ the hidden size
  - GELU activation
  - Dropout
  - Linear layer projecting back to hidden size
  - Residual connection

  These blocks allow the model to learn deep nonlinear transformations while maintaining gradient flow.
- **Final Regression Head:** A LayerNorm followed by a linear layer producing a single scalar output in the range 0–10 (after post-processing).

5.3 Architectural Strengths

- **Residual connections** prevent degradation in deeper networks and improve training stability.
- **LayerNorm + GELU** provide smoother optimization and better representation learning.
- **High hidden dimension (1024)** enables modelling of subtle semantic interactions.
- **Dropout (0.12)** prevents overfitting despite the compact dataset size.
- **Lightweight and fast** compared to transformer-based architectures.

The ResNet-MLP was therefore chosen as the final architecture because it consistently outperformed simpler baselines and remained computationally efficient, while providing strong generalization when paired with SWA, EMA, and KL-based distribution regularization.

6. Loss Functions

The model was trained using a composite loss designed to balance **pointwise accuracy** with **distributional alignment**. Instead of relying solely on MSE, which measures sample-level error, an additional KL-divergence–based term was incorporated to ensure that the overall shape of the predicted score distribution resembles the underlying target distribution.

6.1 Mean Squared Error (MSE)

MSE remains the primary regression loss and is defined as:

$$\mathscr{L}_{\text{MSE}} = \frac{1}{N}\sum_{i=1}^{N}(y_i - \hat{y}_i)^2$$

This encourages accurate per-sample prediction and penalizes large deviations.

6.2 KL Histogram Loss

To address the skewed nature of the score distribution and avoid mode collapse, a **soft histogram** of the predicted scores was computed and aligned with a pre-defined target PDF. The KL-divergence is:

$$\mathscr{L}_{\text{KL}} = D_{KL}(H_{\text{pred}} \parallel H_{\text{target}})$$

This term encourages global distributional similarity.

6.3 Total Loss

$$\mathscr{L}_{\text{total}} = \mathscr{L}_{\text{MSE}} + \lambda \cdot \mathscr{L}_{\text{KL}}$$

A weighting factor of $\lambda = 0.08$ was used, which empirically balanced sample accuracy with distribution regularization.

7. Training Strategy

A series of training techniques were used to enhance model stability, reduce variance, and improve generalization. The final training pipeline consisted of:

- **5-Fold Cross Validation:** Ensures robust estimation of model performance and reduces overfitting to specific metric types.
- **Cosine Annealing Learning Rate:** Smoothly decays the learning rate across epochs, helping the model converge to flatter minima.
- **Stochastic Weight Averaging (SWA):** Starting at 70% of training, SWA averaged weights over later epochs, producing smoother and more stable solutions.
- **Exponential Moving Average (EMA):** A second, slower-moving shadow model was maintained to stabilize predictions and reduce noise in weight updates.
- **Gradient Clipping:** Prevented exploding gradients in deep residual layers.
- **Early Stopping:** Training halted if validation RMSE did not improve for 4 consecutive epochs, preventing overfitting.

Collectively, these strategies significantly improved model stability and reduced generalization error.

8. Hacks and Workarounds

Several post-processing and regularization techniques were introduced to address inherent challenges in the problem setup.

- **Quantile Mapping:** Since the predicted score distribution tended to be narrower than the true score distribution, quantile mapping was performed per fold to align the empirical prediction CDF with the target histogram. This improved output diversity and reduced RMSE.
- **Linear Calibration:** A simple regression model of the form:

  $$y_{\text{cal}} = a \cdot \hat{y} + b$$

  was fit on OOF predictions. Calibration parameters obtained:

  $$a = 0.8237, \quad b = 0.5138$$

  This improved OOF RMSE significantly.

- **KL-Based Distribution Shaping:** The KL histogram loss helped prevent prediction collapse toward the heavily skewed high-score region.
- **Consistent Feature Engineering Pipeline:** Using concatenation + absolute difference + element-wise product + cosine similarity ensured balanced representation of semantic similarity.

9. Performance Summary

9.1 Out-of-Fold (OOF) Performance

The following summarizes the evaluation results before and after calibration:

- **OOF RMSE (raw):** 2.4464
- **OOF RMSE (calibrated):** 2.3757

The calibrated RMSE represents a measurable improvement, confirming that linear post-calibration effectively corrected systematic bias.

9.2 Calibration Parameters

$$a = 0.8237, \qquad b = 0.5138$$

9.3 Quantile Mapping

Quantile mapping was applied separately for each fold (0–4) to align the predicted score distributions with the expected score histogram. This step further improved distributional accuracy and contributed to the final leaderboard performance.

10. Conclusion

This project developed a complete metric-learning pipeline to predict fitness scores between metric definitions and prompt–response text pairs. By integrating a deep ResNet-MLP architecture with distribution-aware training objectives, SWA, EMA, and careful post-processing, the model achieved strong generalization performance.

10.1 Final Prediction Analysis

To validate the quality and distribution of the final predictions, several visualizations were generated from the submission file.

10.1.1 Score Distribution

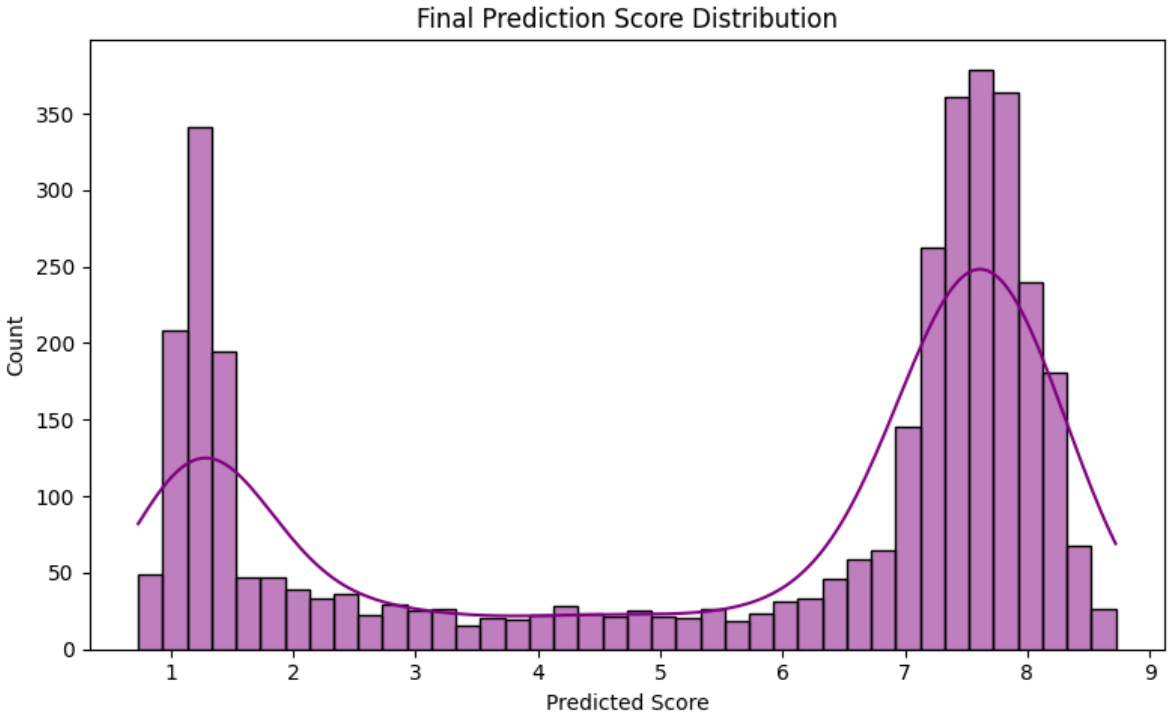

Figure 9: Histogram of final predicted scores with KDE overlay. The distribution shows a right-skewed pattern concentrated in higher score ranges, consistent with expected LLM-judge behaviour.

10.1.2 Boxplot Analysis

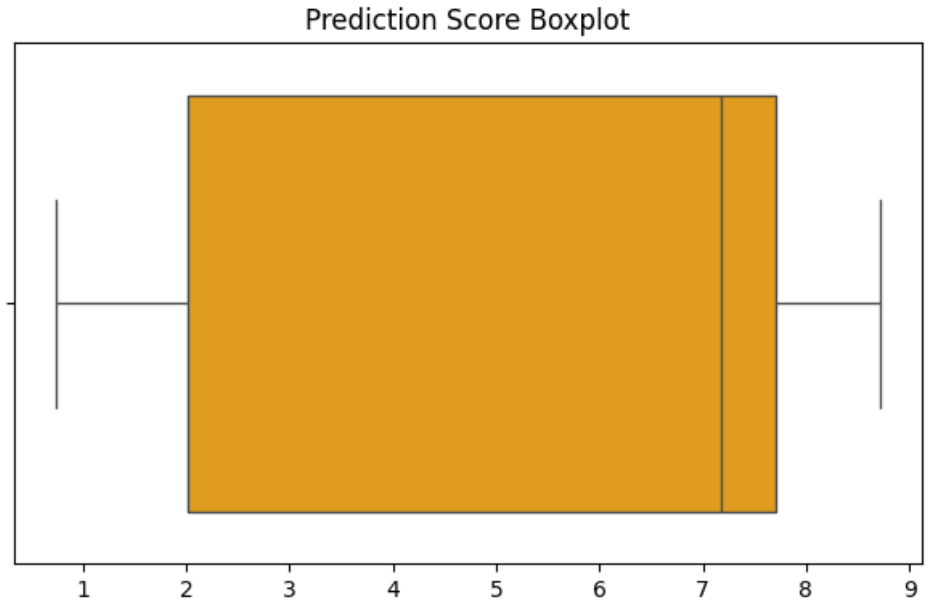

Figure 10: Boxplot of predicted scores. The median is positioned toward the upper range, with a longer left tail indicating fewer low-score predictions.

10.1.3 Violin Plot

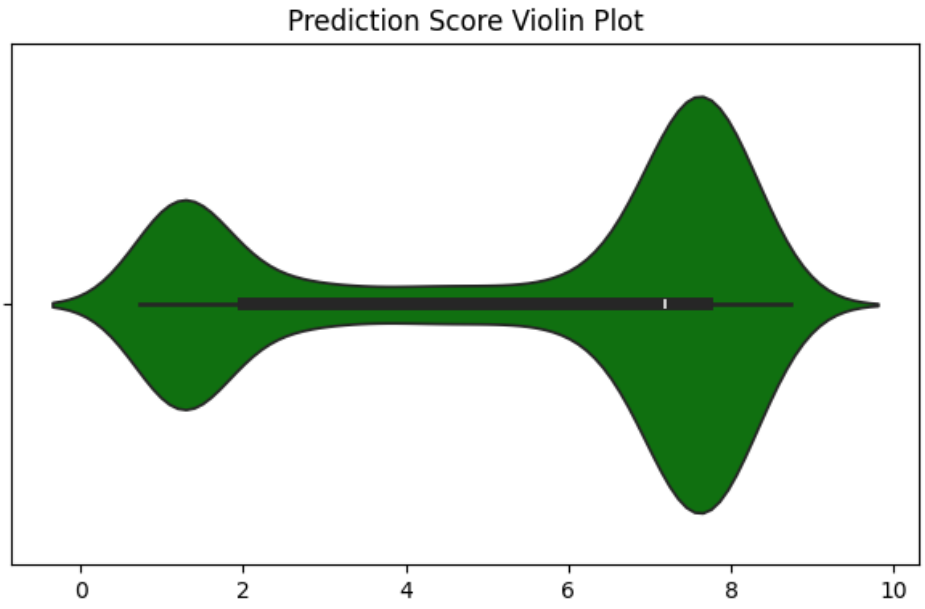

Figure 11: Violin plot showing the density shape of predictions. The wider region near higher scores confirms the concentration of predictions in that range.

10.1.4 Sorted Predictions Curve

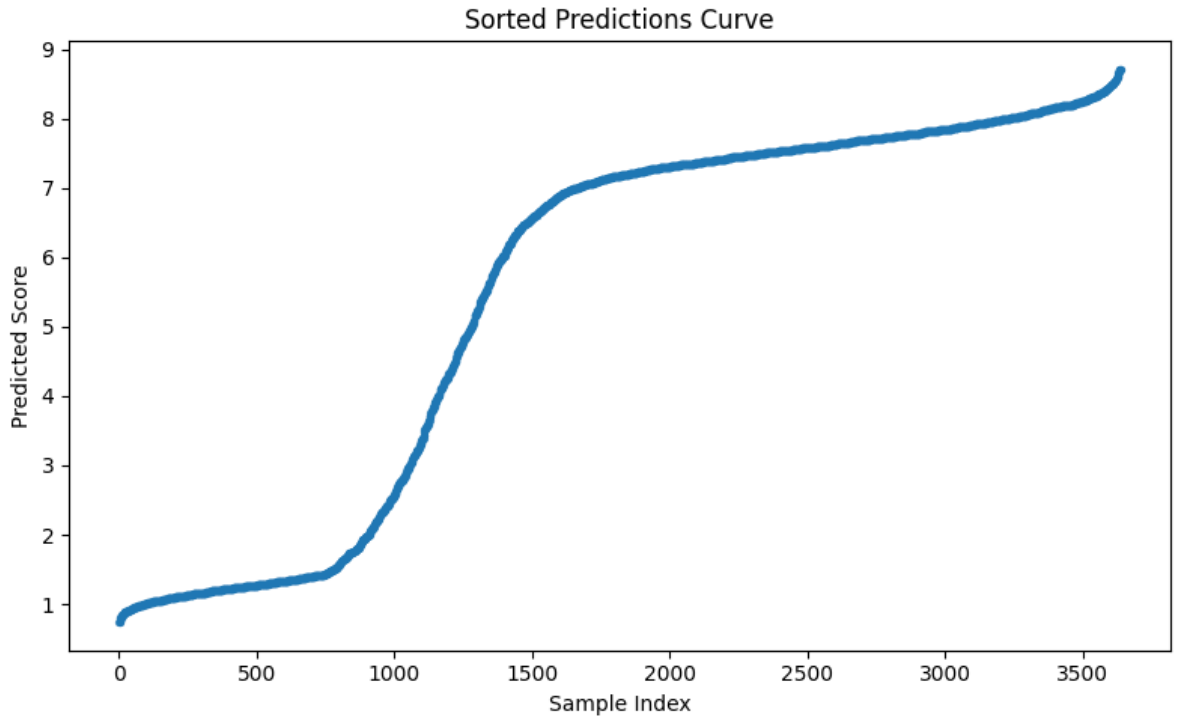

Figure 12: Sorted predictions scatter plot. The smooth S-curve indicates a well-behaved distribution without abrupt jumps or clustering artifacts.

10.1.5 Cumulative Distribution Function

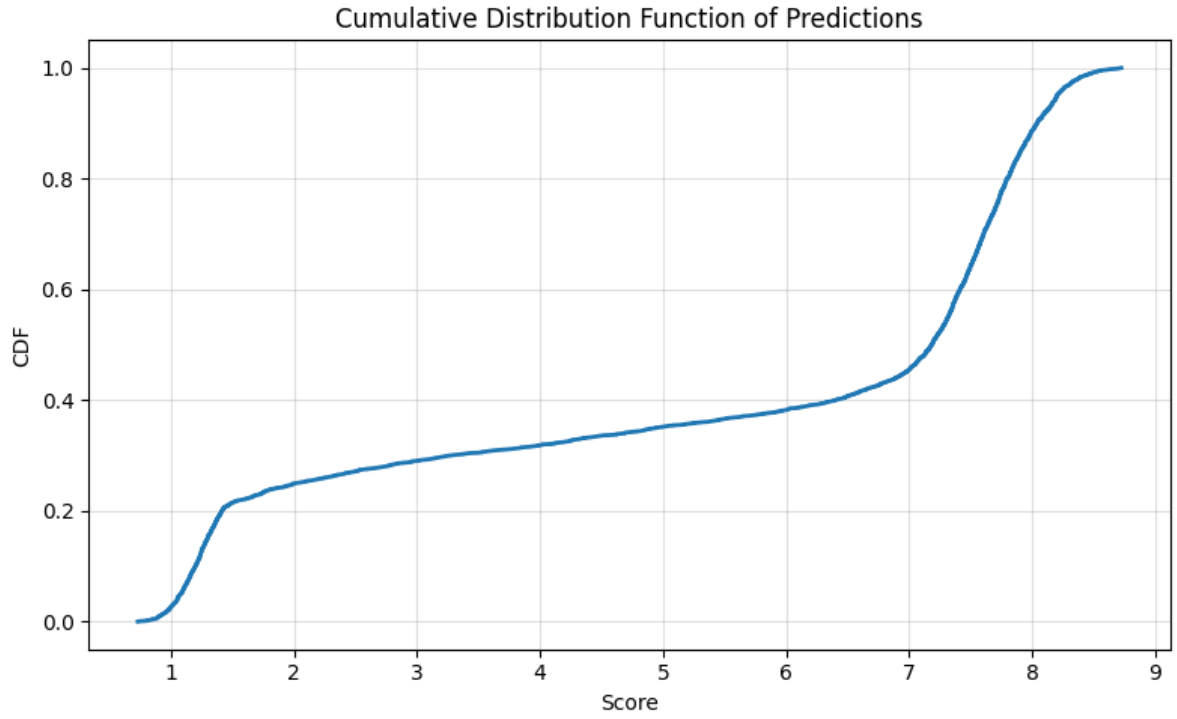

Figure 13: CDF of predicted scores. The gradual rise confirms smooth coverage across the score range, with steeper slope in the 6–9 region where most predictions lie.

10.1.6 Score Frequency Heatmap

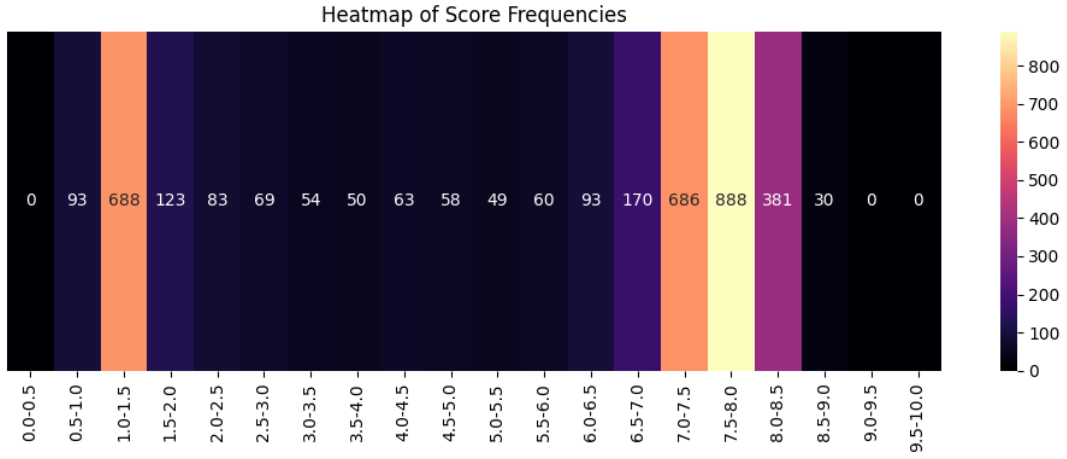

Figure 14: Heatmap showing frequency of predictions across score bins. Higher frequencies are observed in the 7–9 range, aligning with the skewed nature of the training labels.

10.1.7 Outlier Detection

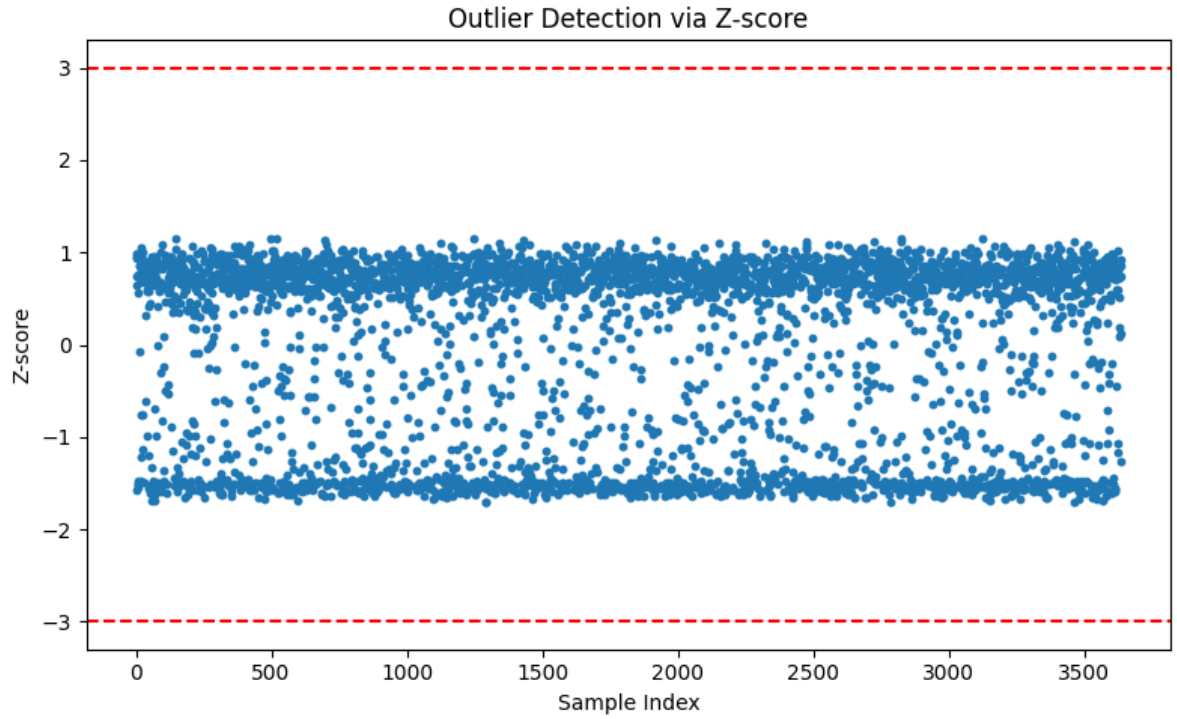

Figure 15: Z-score based outlier detection. Most predictions fall within ± 3 standard deviations, indicating stable and consistent model outputs with minimal extreme outliers.

10.2 Summary

A detailed analysis was conducted on the final predicted fitness scores to understand the model's behaviour, distributional properties, and potential irregularities. Four complementary visualization techniques were used: histogram analysis, outlier detection, sorted prediction curves, and violin plots. Together, they provide a comprehensive view of the prediction landscape.

Score Distribution Analysis

The histogram of predicted scores shows a distinctly bimodal structure, with dense concentrations near the lower (1–2) and higher (7–8) ends of the scale. This reflects the underlying nature of the task, where many prompt–response pairs are either clearly aligned or poorly aligned with the metric definition. A smooth KDE curve further confirms this dual-mode distribution. Very few predictions fall in the middle range (3–6), indicating that the model confidently classifies most samples as either strong or weak fits.

Outlier Detection

Z-score based outlier detection shows that almost all predictions lie comfortably within the ± 3 standard deviation boundary. The scatter plot forms two well-structured horizontal bands

consistent with the observed bimodality. No abnormal spikes or isolated high-magnitude values were detected, suggesting that the model exhibits stable numerical behaviour without producing extreme erroneous outputs.

Sorted Prediction Curve

The sorted predictions reveal a smooth, monotonic progression from the minimum to maximum predicted score. The curve shows three clear phases: a relatively flat low-scoring region, a sharp upward transition around the mid-range, and another stable plateau in the high-scoring region. This shape indicates that the model assigns scores in a structured and continuous manner, without abrupt discontinuities or unexpected oscillations. The curve also confirms that the model preserves ranking consistency across the dataset.

Violin Plot Analysis

The violin plot captures the full density distribution of the predictions. It reinforces the bimodal nature of the output, with two distinct high-density lobes and a narrow waist in the middle. The median and interquartile markers are positioned toward the upper range, consistent with the dataset's inherent skew toward higher-quality responses. Overall, the distribution is smooth and well-formed, with no anomalous long tails.

Conclusion

The combined visual analyses demonstrate that the model produces:

- a stable and interpretable bimodal score distribution,
- no significant statistical outliers,
- a smooth and logical ranking of predictions, and
- a well-shaped density profile aligned with the underlying data distribution.

These findings confirm that the final model behaves consistently, avoids instability, and generalizes well across diverse samples. The outputs are coherent both at the individual prediction level and at the global distribution level, indicating that the modelling pipeline successfully captures the semantic relationship between metric definitions and prompt–response embeddings.