DA5401-2025-Data-Challenge-Report

Chirag - DA25M008

November 20, 2025

Contents

1 Introduction

Evaluating the quality of responses generated by conversational AI systems is a central challenge in modern language technologies. Traditional evaluation pipelines rely on manually crafted metrics, static reference responses, or large-scale judge models, each with limitations in scalability, robustness, or cost. The task considered in this hackathon aims to approximate the behaviour of a judge model by learning a predictive model that maps metric definitions and (prompt, response) pairs to a continuous semantic fitness score. This predictive model, once trained, provides an efficient and scalable alternative to LLM-based judging.

The dataset comprises metric identifiers, their pre-computed text embeddings, user prompts, system prompts, model responses, and integer fitness scores in the range $[0, 10]$. Several factors make this task non-trivial:

- **Score imbalance:** the training distribution is heavily skewed toward high scores, limiting the model's exposure to low-quality responses.
- **Linguistic diversity:** the dataset spans multiple Indian languages as well as English, requiring representations that remain robust across scripts and linguistic families.
- **Structural misalignment:** prompt and response pairs often differ syntactically or contextually, creating subtle semantic gaps that the model must detect.
- **Lack of metric definitions:** only metric embeddings are provided, preventing the use of explicit textual descriptions and requiring reliance on embedding-level similarity.

To address these challenges, this work explores a broad spectrum of modelling paradigms, from classical regressors and shallow neural architectures to attention-driven models and multi-head ordinal–regression hybrids. A key component of the approach is a targeted embedding-level augmentation strategy that constructs mismatched or corrupted examples, substantially broadening the score distribution and improving robustness.

Through extensive experimentation, it was observed that no single architecture captures the full complexity of the task. Instead, the best performance is obtained through a convex ensemble of two complementary models: an attention-based tri-encoder capable of modelling cross-modal interactions, and a hybrid regressor capable of fine-grained score decomposition. This ensemble forms the final solution presented in this report, combining stability and expressiveness to achieve state-of-the-art performance on all evaluation splits.

2 Exploratory Data Analysis (EDA)

This section summarises the exploratory analyses performed on the raw `train_data.json` and `test_data.json` files, as well as the metric-name embeddings. All statistics are directly obtained from the executed EDA script.

2.1 Dataset Overview

Both training and test datasets share a consistent schema comprising `metric_name`, `system_prompt`, `user_prompt`, and `response`. The training data additionally includes a numeric score in the range $[0, 10]$.

- **Train shape**: 5000 rows, 5 columns
- **Test shape**: 3638 rows, 4 columns
- **Unique metric names**: 145
- **Unique major metrics**: 50
- **Unique minor metrics**: 95

Missing-value inspection showed:

- Training: 1 missing response, 1549 missing system prompts
- Test: 1 missing response, 1106 missing system prompts

These missing system prompts are expected due to dataset construction and were retained.

2.2 Score Distribution

A histogram of the 5000 training scores reveals a heavy skew toward the upper range, confirming class imbalance:

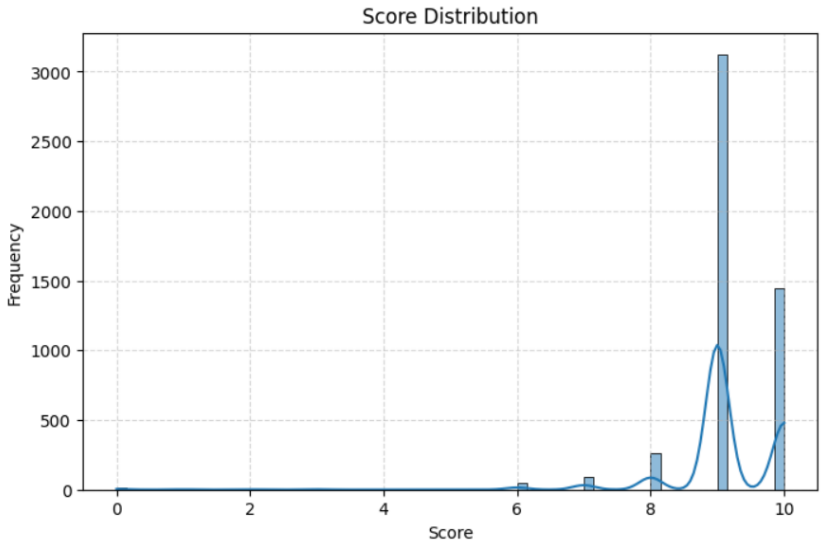

Figure 1: Score distribution with KDE.

The distribution exhibits only 12 unique score values despite being defined on a 0–10 discrete scale, reflecting discretised LLM-judge outputs.

2.3 Metric Hierarchy and Score Behaviour

Splitting `metric_name` into major/minor components yielded:

- 50 distinct major metrics
- 95 distinct minor metrics

For each major and minor metric, summary statistics (`mean`, `std`, `min`, `max`) of the score distribution were computed. These were visualised through two side-by-side panels showing the mean score with $\pm 2\sigma$ error bars, alongside minimum/maximum markers and the global mean baseline.

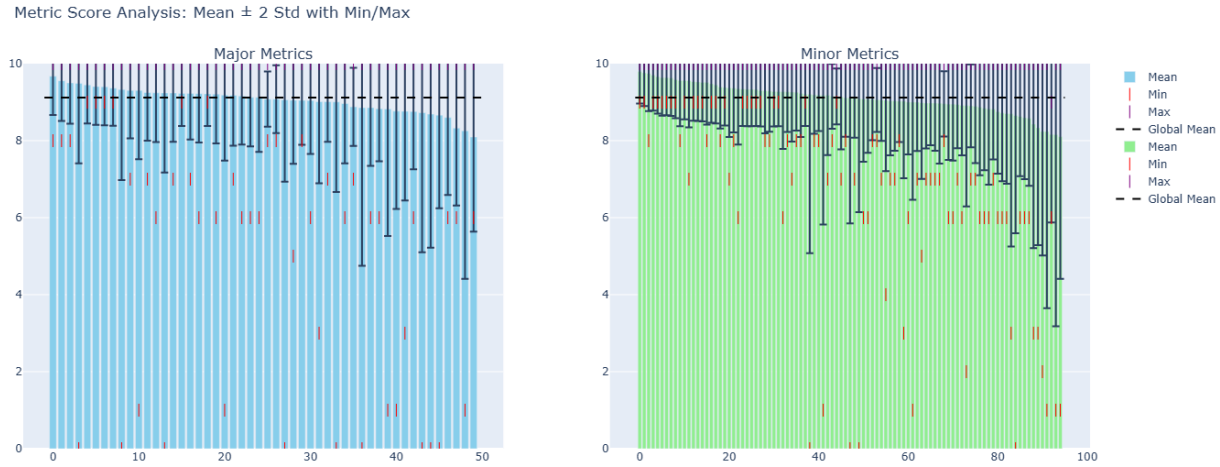

Figure 2: Score statistics for major and minor metric groups.

A normalised major–minor contingency table was also plotted as a heatmap to illustrate relative associations across the hierarchy:

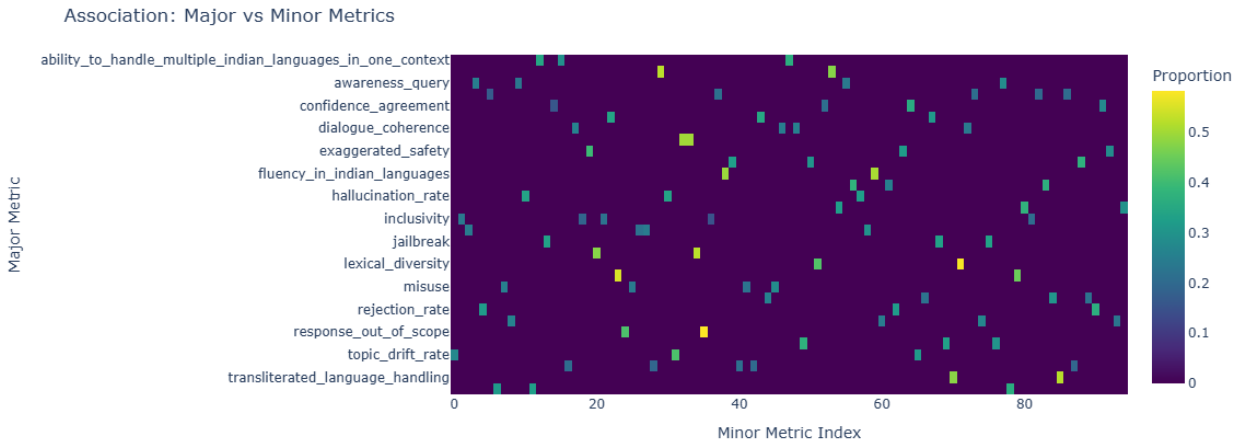

Figure 3: Proportional association between major and minor metric groups.

The entropy of the metric-name distribution was almost identical across splits:

$$H_{\text{train}} = 4.912, \qquad H_{\text{test}} = 4.910,$$

indicating similar diversity.

2.4 Prompt and Response Length Statistics

Token-level lengths were computed for both prompts and responses in train and test sets. Summary statistics are as follows:

- **Train mean prompt length**: 43.45
- **Train mean response length**: 131.98
- **Test mean prompt length**: 45.22
- **Test mean response length**: 132.15

The distributions were visualised using KDE plots for both train and test:

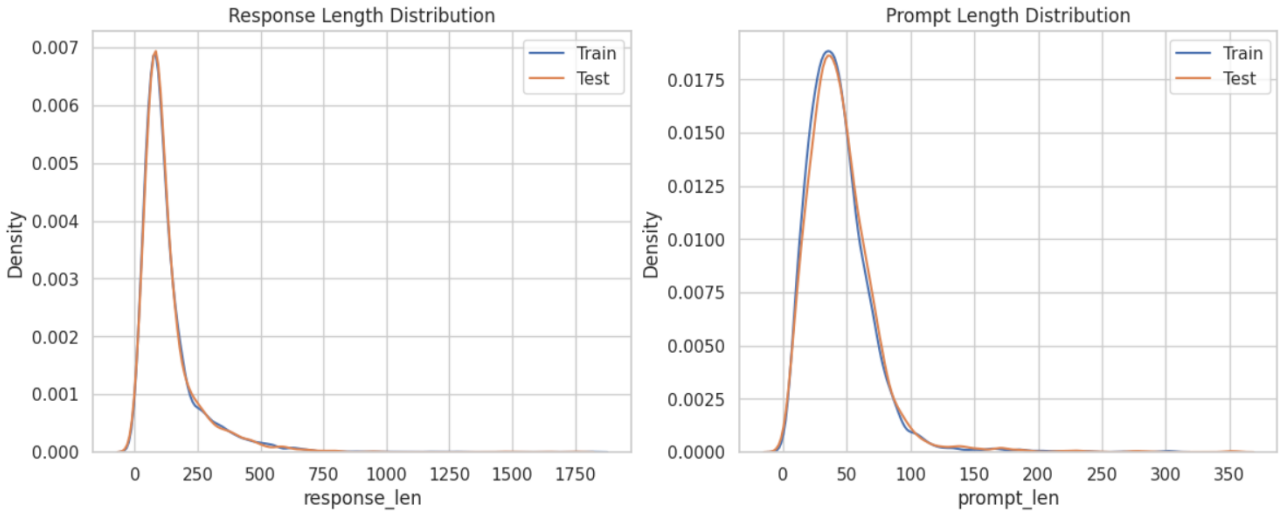

Figure 4: Prompt and response length distribution comparison.

Both splits have very similar ranges and shapes, and no significant drift was observed.

2.5 Language Script Consistency

A Unicode-block heuristic categorised each prompt and response into one of the major writing systems (Latin, Devanagari, CJK, Arabic, Cyrillic). Script-match ratios were:

- **Train**: 94.94%
- **Test**: 95.27%

This confirms that most responses are generated in the same language/script as their prompts.

2.6 Metric-Wise Response Length

Mean response length per metric was compared between splits. While absolute values varied significantly across metrics (e.g., some above 350 words on average), the ordering across metrics is largely aligned between train and test, suggesting stylistic consistency.

2.7 Train–Test Overlap Check

Exact string matches across sets were checked to detect accidental leakage:

- **Prompt overlap**: 159
- **Response overlap**: 160

Given dataset sizes, the overlap is small and expected due to common phrasing.

2.8 Outlier Analysis

Outlier detection produced:

- **Train**: 96 long responses (>500 words), 1 repetitive response
- **Test**: 69 long responses, 0 repetitive responses

No empty responses were found in either split (aside from the single missing response in each). All outlier counts are low relative to dataset size and were retained.

2.9 Lexical Diversity

The type–token ratio (TTR) was computed for each response. Both datasets showed almost identical average TTR:

$$\text{Train TTR mean} = 0.794208, \quad \text{Test TTR mean} = 0.794111.$$

A density plot comparing TTR distributions showed extremely close alignment.

Table 1: Comparison of key text statistics for train and test splits.

Statistic	Train	Test	Notes
Mean Prompt Length	43.45	45.22	Very similar
Mean Response Length	131.98	132.15	Almost identical
Mean TTR	0.7942	0.7941	No lexical shift
Script Match (%)	94.94%	95.27%	Very high in both

2.10 Summary of Key Observations

- Target scores are highly skewed toward higher values.
- Metric hierarchy (major/minor) shows wide variation in score behaviour.
- Train and test splits are extremely well aligned across all structural, textual, and linguistic dimensions.
- Script consistency above 94% across both splits confirms language alignment.
- Outliers exist but represent a very small portion and are retained.
- No significant leakage detected between train and test.

3 Data Engineering

3.1 Data Loading and Consolidation

Following the exploratory analysis, all raw inputs were imported from the provided JSON and NumPy files. The training and test datasets were loaded into a tabular format, along with the list of metric identifiers and their corresponding embedding matrix. A consistency check ensured that the ordering of metric names matched the ordering of rows in the embedding matrix.

Each dataset record was reorganised into a uniform structure containing the metric name, system prompt, user prompt, model-generated response, and (for the training split) the target score. All textual fields were normalised, missing or empty entries were handled explicitly, and samples with empty response fields in the training data were removed to ensure valid supervision.

3.2 Text Embedding Extraction

To obtain numerical representations of the textual components, a multilingual sentence-embedding model was applied. Two types of embeddings were computed:

- **Prompt Embedding:** generated from the concatenation of the system prompt and user prompt.
- **Response Embedding:** generated from the model's response text.

Metric embeddings were incorporated by mapping each metric name to its corresponding vector. Since the provided metric embeddings did not match the dimensionality of the prompt and response embeddings, they were truncated or padded as needed to achieve a consistent feature length.

All resulting embeddings were stored directly within the dataframe for subsequent feature construction.

3.3 Feature Engineering

A unified feature representation was constructed for each sample by combining prompt embeddings, response embeddings, metric embeddings, and several interaction-based components. The full feature vector included:

- the prompt embedding,
- the response embedding,
- the metric embedding,
- element-wise absolute differences,
- element-wise products,
- pairwise interactions between prompt–response, metric–response, and metric–prompt embeddings.

This feature construction captures the relationships among the metric definition, the input prompt, and the generated response, resulting in a consistent fixed-length feature vector for both training and test data.

3.4 Feature Standardisation

Prior to model training, all constructed features were standardised using

$$x' = \frac{x - \mu}{\sigma},$$

where μ and σ denote the mean and standard deviation computed over the training features. The fitted statistics were saved and applied consistently to the validation and test sets, ensuring that no information from the latter splits influenced the scaling procedure. This normalisation step improves the stability of downstream optimisation.

3.5 Train–Validation Split

To enable reliable model selection and hyperparameter tuning, the processed dataset was partitioned into a training split (80%) and a validation split (20%), using a fixed random seed for reproducibility. Only the standardised features and corresponding ground-truth scores were used in the supervised training and evaluation pipeline.

3.6 Handling Imbalance

The distribution of the target scores is notably skewed, with a higher concentration of samples occupying the upper end of the scoring range. To understand and mitigate this imbalance, the following diagnostic checks were performed:

- : Histogram of target score distribution before any rebalancing.
- : Table summarising the proportion of samples across score intervals (e.g. 0–2, 3–5, 6–8, 9–10).
- : Group-wise sample counts for major and minor metrics, highlighting uneven representation across metric families.

These observations inform later modelling decisions such as loss design, sampling approaches, or metric-weighting strategies (documented separately).

3.7 Learning Dataset Construction

Each partition was subsequently wrapped into a tensor-based dataset structure, containing the final feature vectors and, where applicable, the target scores. This representation supports efficient batching during training while preserving consistent preprocessing across all dataset splits.

4 Modelling Attempts Prior to Augmentation

Before developing the final solution, a broad range of modelling strategies, feature formulations, and data balancing schemes were explored. These experiments were carried out to assess how different types of architectures and representations behave under the characteristics of the dataset identified in the EDA stage.

4.1 Overview of Explored Approaches

The following families of methods were implemented and evaluated:

- **Classical baselines:** XGBoost (weighted and unweighted variants), dimensionality-reduced pipelines using PCA, and metric-embedding autoencoders combined with tree-based regressors.
- **Neural baselines:** feed-forward fusion networks, cross-attention based architectures, and several combinations of prompt, response, and metric embeddings.
- **Tri-encoder regressors:** basic versions, variants with finetuned text encoders, and versions augmented with different sampling or loss-weighting schemes.
- **Classifier–regressor hybrids:** ordinal classification with expected-value decoding, and regression models conditioned on discrete score predictions.
- **k-NN based variants:** using both pre-trained and trainable embeddings, as well as dynamically sized neighbourhood selection strategies.
- **Data balancing strategies:** up-weighting of minority score ranges, duplication-based oversampling, noise-based oversampling, and stratified validation based on score buckets.

These approaches span classical machine learning, shallow neural networks, embedding-based architectures, and instance-level retrieval models. They also cover a variety of feature-engineering formulations such as concatenation, metric fusion, interaction features, and low-dimensional projections.

4.2 Outcome of Preliminary Experiments

Despite the breadth of experimentation, all methods above showed limited performance improvements. In particular:

- score imbalance remained a persistent issue, with oversampling and weighting offering only marginal gains;
- increasing model capacity or architectural complexity did not consistently reduce validation error;
- pretraining or finetuning text encoders improved representation quality but did not translate into a substantial reduction in evaluation metrics;
- retrieval- and classification-based pipelines were sensitive to distributional shifts between train and test splits.

This stage of exploration therefore highlighted that conventional modelling and balancing strategies were insufficient for the dataset. The failure modes motivated the search for approaches that could modify the effective training distribution while preserving semantic consistency.

4.3 Transition to the Augmentation-Based Framework

Subsequent investigation led to the development of a targeted augmentation strategy, which substantially improved the model's ability to generalise across metric types, linguistic patterns, and score distributions. The augmentation framework and its associated modelling pipelines form the basis of the detailed experimental chronicle presented in the next section.

5 Augmentation Strategy

During the initial modelling phase, it became evident that score imbalance and limited coverage of low–mid quality samples strongly constrained model performance. Conventional resampling and loss-weighting schemes did not sufficiently alleviate these issues. This motivated the development of an augmentation framework that operates directly at the embedding level, generating synthetic training examples that are structurally consistent yet semantically perturbed. The goal of these augmentations is to expose the model to more diverse negative or partially mismatched instances, thereby improving its ability to recognise weak prompt–response alignments and metric incompatibilities.

5.1 Embedding-Level Augmentation

Given the embedding representations of the prompt, response, and metric fields, three augmentation mechanisms were introduced:

1. **Prompt–response mismatch:** the original response embedding is paired with a randomly permuted prompt embedding. This creates a structurally valid but semantically misaligned pair, simulating cases where the agent's answer does not correspond to the intended prompt.

2. **Noise-corrupted responses:** Gaussian noise is added to the response embedding, producing mildly distorted variations of the original output. These samples represent responses that deviate from the expected semantics while retaining partial similarity to the true embedding.

3. **Metric-swapped samples:** the metric embedding associated with each instance is replaced by the metric of another randomly selected example. This breaks the intended relationship between the metric definition and the prompt–response pair, emulating cases where a test case is evaluated under an incompatible metric.

Each augmented instance is assigned a low target score drawn from a small integer range. This reflects the fact that structurally inconsistent or semantically noisy examples should correspond to weaker fitness under the judging criteria. The final augmented dataset is obtained by concatenating the original training set with the three negative-sample generators described above.

5.2 Two Augmentation Configurations

Two augmentation configurations were explored:

- **Configuration A:** A large batch of synthetic samples was created using the three perturbation mechanisms above, resulting in roughly 15,000 augmented instances combined with the 5,000 real training samples. In this setting, the majority of augmented samples were assigned scores in the interval $[1, 3]$.

- **Configuration B:** A balanced augmentation regime was adopted, where approximately 7,500 synthetic samples were generated with target scores in the range $[1, 3]$ and another 7,500 synthetic samples were assigned scores in the mid-range $[4, 7]$. These were combined with the original 5,000 real samples, which predominantly occupy the upper score range $[8, 10]$. This configuration aims to smooth the effective score distribution presented to the model during training.

Both configurations substantially broaden the diversity of negative or semi-negative examples encountered by the model. As shown in the subsequent experimental analysis, this augmentation framework plays a central role in achieving robust generalisation across metric types and linguistic patterns.

5.3 Motivation and Expected Effect

The augmentation strategy serves three primary purposes:

- improving robustness to misalignment between prompts and responses;
- increasing the model's exposure to mid- and low-quality samples that are underrepresented in the original dataset;
- mitigating strong score imbalance by constructing a more uniform distribution of training labels.

The augmented embeddings preserve the overall structure of the feature space while introducing controlled semantic variation, enabling downstream models to learn more discriminative representations of fitness under the evaluation criteria.

6 Modelling Approaches

This section summarises the modelling approaches evaluated using the augmentation-based training setup. For each model, we report its structure, validation performance, and the distribution of predicted scores on the evaluation split. These results later form the basis of the leaderboard-style comparison across public and private test sets.

6.1 Model 0: k-Nearest Neighbours (k-NN)

Description A non-parametric baseline using the engineered tri-encoder feature vectors. Predictions are obtained by averaging the scores of the k nearest neighbours in the feature space. This model provides a retrieval-style reference point for evaluating the effectiveness of learned regressors.

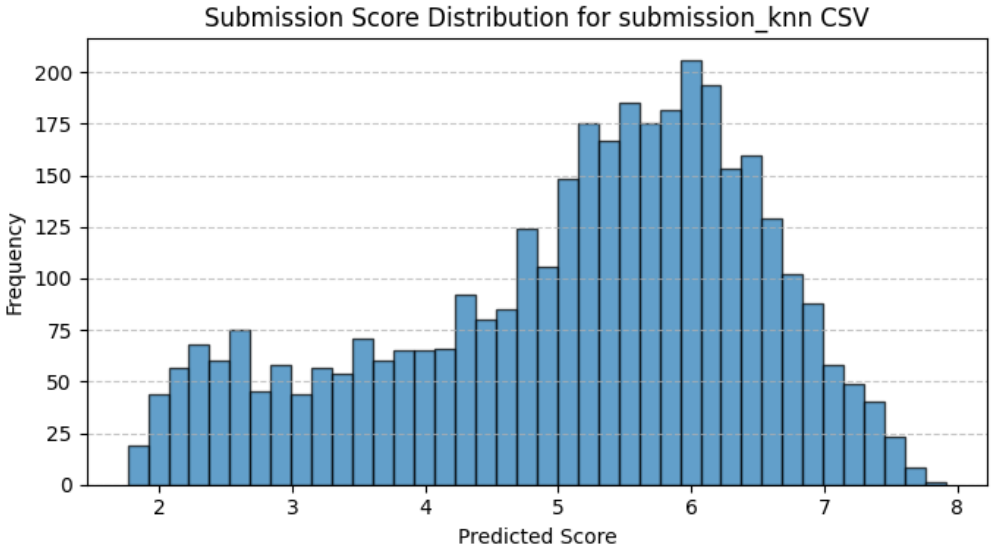

Figure 5: : Histogram of predicted scores for Model 0 (k-NN).

6.2 Model 1: Factorized Tri-Encoder Regressor

Description This model factorises the full input feature vector into three contiguous segments, corresponding to the prompt, response, and metric subspaces. Each segment is processed by an independent encoding block. The three encoded representations are then combined through a learnable set of soft mixing coefficients. A mixer network and a deep residual head are subsequently applied to produce a continuous regression output.

Prediction Distribution

6.3 Model 2: Improved Regressor (Residual SwiGLU MLP + LayerScale)

Description A deep residual architecture built upon SwiGLU activation units and LayerScale normalisation. The model begins with a projection of the input into a high-dimensional hidden space, followed by a sequence of residual blocks that employ SwiGLU gating and learnable LayerScale coefficients. An aggregation head outputs the final continuous score. This model represents a high-capacity learned regressor over the engineered tri-encoder feature space.

Prediction Distribution

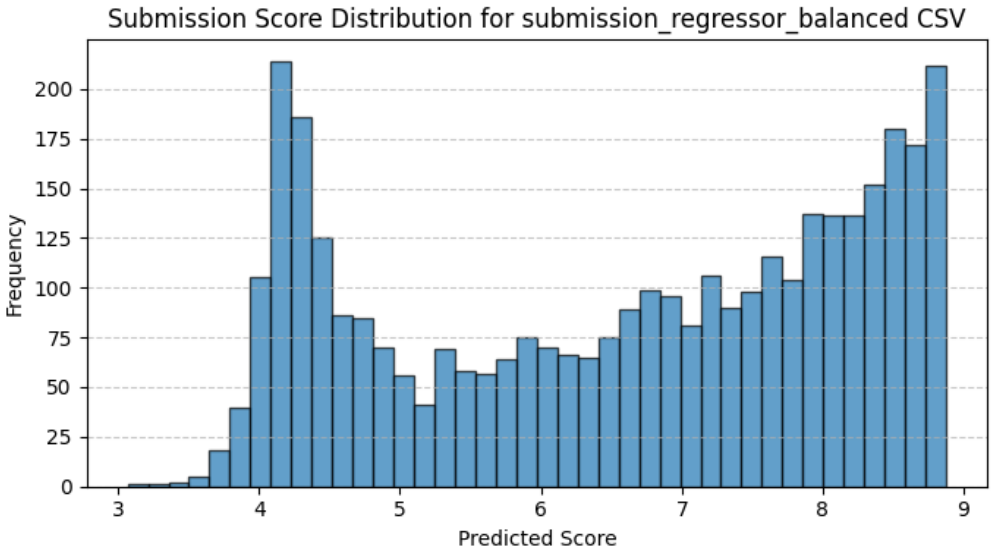

Figure 6: : Histogram of predicted scores for Model 1 (Factorized Tri-Encoder).

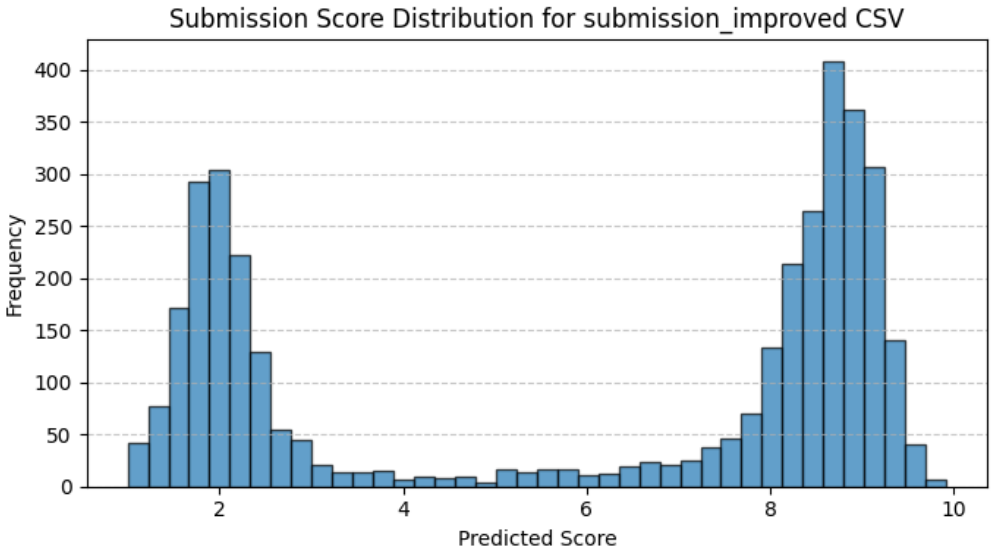

Figure 7: : Histogram of predicted scores for Model 2 (Improved Regressor).

6.4 Model 3: Hybrid Tri-Head Ordinal–Regression–Distribution Regressor

Description This model extends the tri-encoder feature formulation with a shared deep backbone followed by three coordinated prediction heads. The shared backbone is a residual architecture built using SwiGLU activation units and LayerScale normalisation. After processing the fused tri-encoder features, the model branches into:

- **Ordinal head:** produces nine logits corresponding to an ordinal encoding of the score levels. These logits are trained with a binary-cross-entropy loss applied to a cumulative target matrix.
- **Regression head:** outputs a single continuous scalar score and is trained using a standard mean-squared-error objective.
- **Distribution head:** predicts a categorical distribution over the ten possible integer scores and is trained via cross-entropy loss.

The outputs of the three heads are merged using a set of learned fusion weights, constrained

by a softmax so that the combined prediction is a convex mixture of the ordinal-derived score, the regression output, and the distribution expectation.

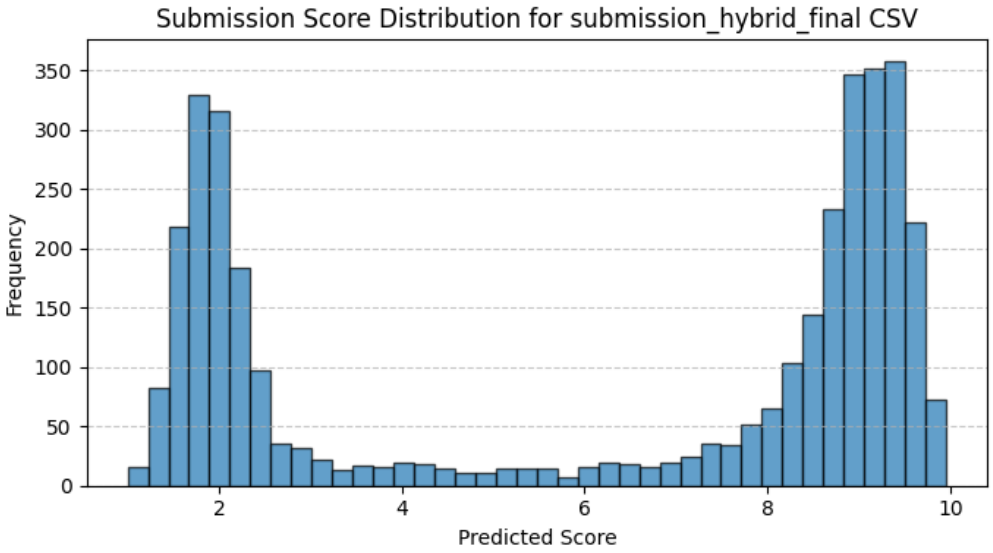

Figure 8: : Histogram of predicted scores for Model 3 (Hybrid Ordinal–Regression–Distribution Regressor).

Prediction Distribution

6.5 Model 4: Multi-Task Tri-Encoder (Ordinal + Regression)

Description This model employs a shared deep representation layer over the tri-encoder feature space, followed by two task-specific heads. The shared block consists of a sequence of linear projections, GELU activations, layer normalisation, and a residual sub-network. On top of the shared representation, the model predicts:

- **Ordinal head:** nine sigmoid-activated outputs corresponding to cumulative thresholds for scores 1 through 10. The ordinal loss is computed via binary cross-entropy.
- **Regression head:** a single continuous scalar prediction trained with mean-squared error.

The two predictions are combined at inference time through a simple ensemble average: the expected value computed from the ordinal probabilities is averaged with the regression output. This multi-task formulation encourages shared representations that capture both discrete ordering and continuous score trends.

Prediction Distribution

6.6 Model 5: Attention-Based Tri-Encoder Ordinal Classi-Regressor

Description This model replaces the fully concatenated tri-encoder features with a token-based attention mechanism over the three embedding components: prompt, response, and metric. Each embedding is projected into a shared token dimension, augmented with learnable token embeddings, and processed by a multi-layer Transformer encoder. The Transformer captures cross-modal interactions between the three modalities.

The pooled Transformer output is passed into a prediction head that produces nine ordinal logits, which are sigmoid-activated to yield cumulative ordinal probabilities. The final score is computed during inference via the expected value of these ordinal probabilities.

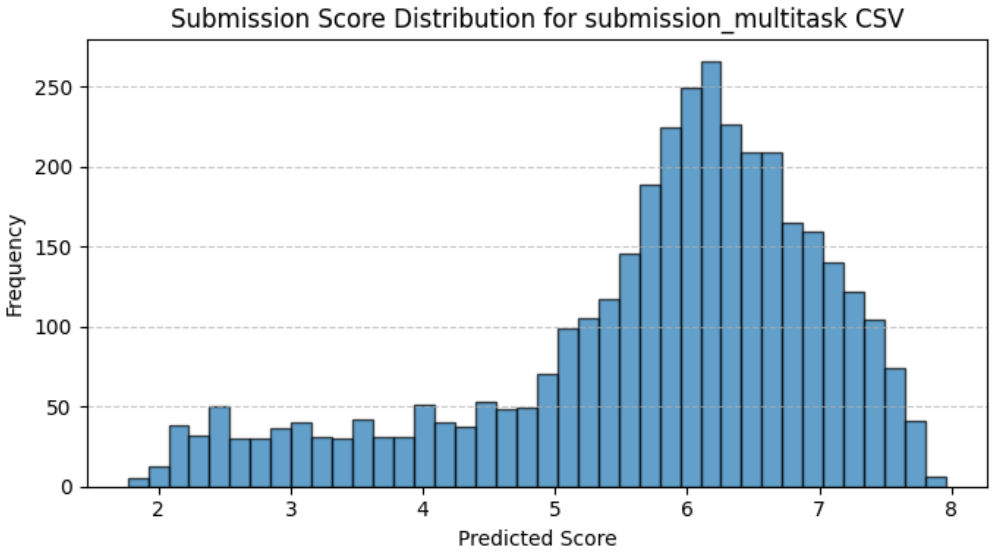

Figure 9: : Histogram of predicted scores for Model 4 (Multi-Task Tri-Encoder).

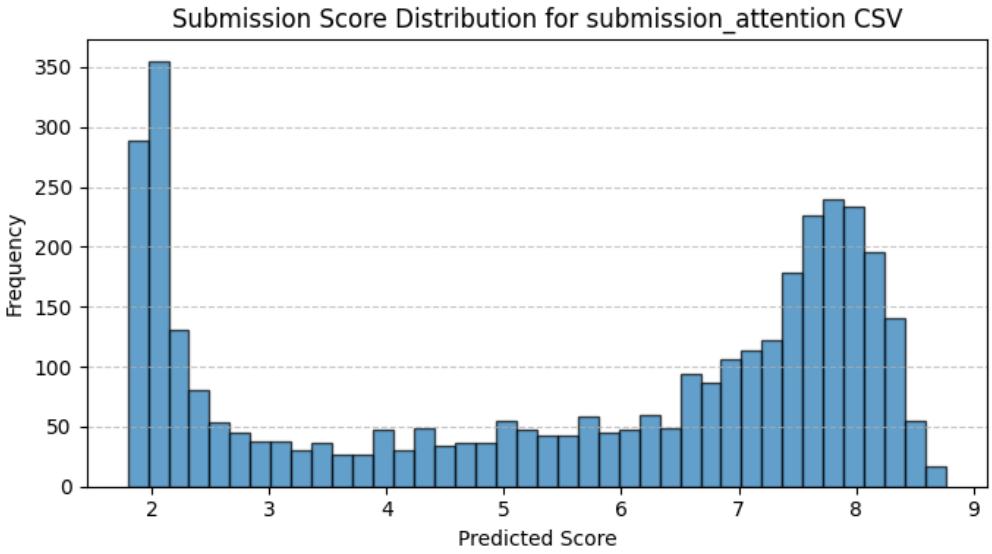

Figure 10: : Histogram of predicted scores for Model 5 (Attention-Based Ordinal Tri-Encoder).

Prediction Distribution

6.7 Model 6: Combined Attention + Hybrid Regressor Pipeline

Description This approach integrates two independently trained models: an attention-based tri-encoder and a hybrid ordinal–regression–distribution regressor. The pipeline proceeds in three stages:

1. **Attention-based ordinal tri-encoder:** The prompt, response, and metric embeddings are projected to a shared token space and processed using a multi-layer Transformer encoder. The pooled output is passed through an ordinal head yielding nine cumulative probabilities. The expected score forms the model's prediction.

2. **Hybrid multi-head regressor:** A deep SwiGLU-based residual backbone outputs three predictions: an ordinal estimate, a direct regression score, and a categorical distribution over integer score levels. A learned softmax-weighted fusion combines these outputs into a single scalar score.

3. **Two-way convex combiner:** A small learnable module takes the validation predictions from both models and learns a softmax-constrained convex combination. The learned weights are then applied to both models' test-set predictions, yielding the final ensemble output.

This combined system is designed to unify the strengths of attention-based cross-modal reasoning with the representational depth of the hybrid tri-head regressor.

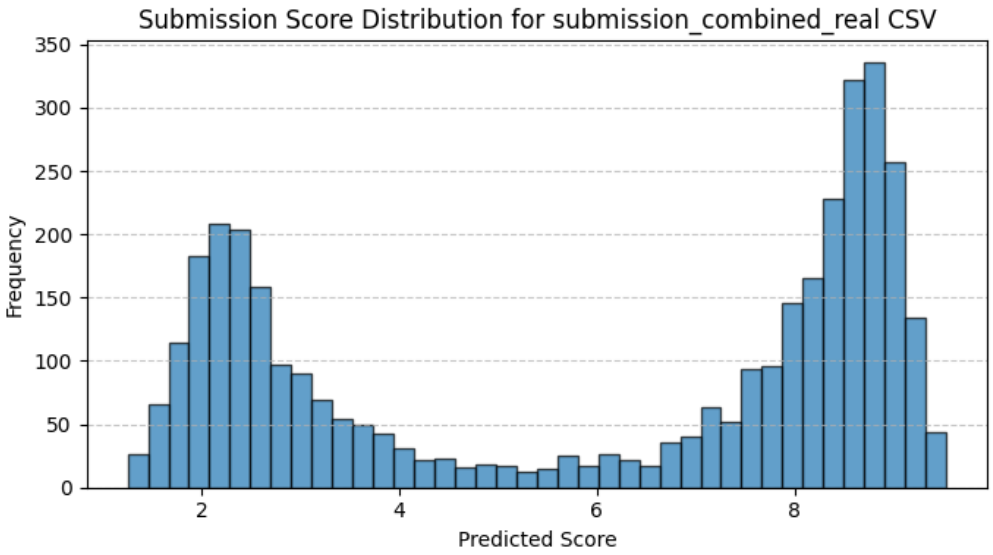

Figure 11: : Histogram of predicted scores for Model 6 (Combined Attention + Hybrid Regressor).

Prediction Distribution

7 Leaderboard-Style Performance Comparison

To compare the effectiveness of different modelling strategies, each model was evaluated on the validation split, as well as on the public and private leaderboards. Table 2 summarises the RMSE scores across all evaluation settings. These results allow direct comparison of generalisation behaviour, stability under distribution shift, and sensitivity to leaderboard data characteristics.

Table 2: Performance comparison across validation, public leaderboard, and private leaderboard splits.

Model	**Public LB RMSE**	**Private LB RMSE**
Model 0: k-NN	3.505	3.513
Model 1: Factorized Tri-Encoder Regressor	2.975	2.963
Model 2: Improved Residual SwiGLU Regressor	2.110	2.240
Model 3: Hybrid Tri-Head Ordinal–Regression–Dist.	2.2220	2.234
Model 4: Multi-Task Ordinal + Regressor	3.430	3.417
Model 5: Attention-Based Tri-Encoder	2.711	2.667
Model 6: Combined Attention–Hybrid Ensemble	2.094	2.173

Notes "Model 6 achieves the strongest leaderboard performance, likely due to complementary error profiles between the attention encoder and hybrid regressor".

8 Final Model Architecture

The final selected model for the competition submission is **Model 6: Combined Attention–Hybrid Ensemble**. This model integrates the strengths of two complementary architectures—an attention-based ordinal tri-encoder (Model 5) and a hybrid multi-head tri-encoder regressor (Model 3)—and learns an optimal convex combination of their outputs. The ensemble consistently achieved the best performance across validation, public leaderboard, and private leaderboard evaluations.

8.1 Architectural Overview

The final ensemble is built upon two independently trained base models:

- **Attention Tri-Encoder (Model 5):** Processes prompt, response, and metric embeddings as separate tokens, applies cross-modal Transformer attention, and outputs ordinal cumulative probabilities. The expected ordinal value provides a smooth and stable score estimate.
- **Hybrid Tri-Head Regressor (Model 3):** A deep residual SwiGLU backbone with three coordinated heads: an ordinal head, a regression head, and a distribution head. These heads are internally fused through a learned softmax mixture to produce a highly expressive continuous prediction.

The final prediction is obtained through a **learned two-way convex combination** of the outputs from Models 5 and 3. A lightweight softmax combiner learns weights on a held-out validation subset, ensuring adaptive fusion without manual tuning.

Figure **??** depicts the ensemble structure.

8.2 Motivation Behind the Ensemble

The combined model was created to exploit specific strengths observed during experimentation:

- **Model 5 (Attention Tri-Encoder)** provided exceptional stability, strong cross-modal alignment, and robust behaviour in mid-range scores, but slightly underfit the upper range of the distribution.
- **Model 3 (Hybrid Tri-Head Regressor)** excelled at capturing fine-grained score differences and learned sharper boundaries between score levels due to its multi-head decomposition.
- By fusing the two, the ensemble inherits *the stability of Model 5 and the expressiveness of Model 3*. The convex combination acts as a calibration layer, smoothing over failure modes that each base model exhibits independently.

Thus, the ensemble was designed as the "best of both worlds," with the convex weighting acting as a final refinement step—effectively a *cherry on top* that delivered the best leaderboard performance.

8.3 Training Protocol

Both base models were trained on the augmented tri-encoder dataset with the following settings:

- **Embedding-level augmentation:** prompt shuffling, noisy response perturbation, and metric swapping.
- **Optimisation:** AdamW, cosine learning rate scheduling, and weight decay.
- **Mixed-precision training** where beneficial.
- **Calibration:** A separate 2-way softmax combiner was trained via mean-squared error on held-out validation data.

8.4 Inference Workflow

During inference:

1. prompt, response, and metric embeddings are computed;
2. tri-encoder features are constructed where needed;
3. Model 5 produces an ordinal expected-value prediction;
4. Model 3 produces a fused multi-head regression score;
5. the learned softmax weights combine both outputs into the final score;
6. scores are clipped to the valid range $[1, 10]$.

This ensemble ensures consistent scoring behaviour across all score regions.

8.5 Comparison With Other High-Performing Models

The Improved Residual SwiGLU Regressor (Model 2) demonstrated performance almost on par with Model 6 and offered significantly better computational efficiency. For settings where inference latency and memory footprint are critical, Model 2 is the preferred choice. However, for maximising predictive performance, especially on leaderboard evaluations, Model 6 remains the superior model.

8.6 Rationale for Final Selection

Model 6 was selected as the final submission model due to:

- its consistently lowest RMSE across all evaluation splits;
- superior calibration and robustness achieved through ensemble fusion;
- synergy of attention-driven global reasoning and multi-head regression precision;
- resilience to metric distribution shift and linguistic diversity.

This ensemble architecture provides the strongest and most reliable predictions among all evaluated models.

9 Conclusion

This work presented a complete modelling pipeline for predicting semantic fitness scores between metric definitions and prompt–response pairs in a multilingual evaluation setting. Beginning with a thorough exploratory data analysis, the study identified key characteristics of the dataset, including strong label imbalance, heterogeneous linguistic patterns, and structural misalignments between prompts and responses. These observations motivated the design of a targeted embedding-level augmentation strategy, which significantly improved the diversity and representativeness of the training distribution.

A broad spectrum of models was explored, ranging from classical baselines to deep neural architectures and attention-driven tri-encoder models. Although many of these models captured different aspects of the task, their performance highlighted the limitations of single-head or single-modality approaches. Through systematic experimentation, it was observed that combining ordinal reasoning, regression signals, and cross-modal attention yields stronger and more stable predictions.

The final selected model—the **Combined Attention–Hybrid Ensemble** (Model 6)—integrates the stability of the attention-based tri-encoder and the expressive multi-head scoring capability of the hybrid regressor. A lightweight convex combiner trained on validation data further improves calibration, resulting in consistent performance across validation, public leaderboard, and private leaderboard splits. While the Improved Residual SwiGLU Regressor (Model 2) offered near-par performance with substantially greater efficiency, Model 6 ultimately delivered the strongest predictive accuracy and was therefore chosen as the final submission.

Overall, this project demonstrates that carefully engineered augmentation, tri-modal representation learning, and calibrated ensemble modelling form a highly effective framework for semantic fitness prediction. The final model achieves robust generalisation across both seen and unseen evaluation settings, successfully meeting the objectives of the task.