Report: Kaggle Challenge

Anumita (BE22B004)

Summary of Model

The model uses a **LightGBM regressor** and feature engineering. 4 768-dimensional embeddings: metric definitions, system prompt, response and user prompt were used. The main goal was to make sure the model understood the semantic relationships between these embeddings and could map them to a score between 1-10. Some of the main ideas that i used in the model which helped improve performance include:

- **Augmentation:** The data had very few low scores. To fix this, synthetic negative samples were added. To make sure the model learned perfect scores well, score 10 samples were oversampled with small noise added.
- **Dimensionality reduction:** PCA was applied to global embeddings and interaction features like differences and element-wise products. This made the model faster and helped it learn more stable patterns.
- **Scalar similarity features:** Cosine similarity and L1 and L2 distances were added so the model had simple and direct measures of how similar different embeddings were.

The system was trained with stratified K-fold cross-validation and early stopping to avoid overfitting.

Report

Problem Statement

The dataset contained metric definitions, system and user prompt and response and final quality scores. The goal was to learn a mapping from these embeddings to the score by understanding semantic similarity.

Objectives

- Build a feature-rich pipeline using embedding interactions.
- Train a regression model - used LightGBM
- Validate the approach.

Understanding the Train Data

The dataset contains multilingual conversational data. From this four 768-dimensional embeddings were created for each sample: metric, system prompt, user prompt and response. A score between 0 and 10 was given for each sample. A detailed exploratory data analysis was performed to understand the dataset.

- **Target variable imbalance:** Most scores fall between 8 and 10. The mid-range and low scores are scarce. Because of this imbalance, augmentation was necessary to help the model learn how low scores look.

- **Language variation:** Many samples are in Hindi and English, with some in Bengali and Tamil. This makes multilingual embeddings important for capturing meaning across languages.
- **Response length:** Responses can be very long (sometimes over 12,000 characters). The variability highlights the need for embeddings that can compress large text inputs.
- **Metric embeddings:** PCA plots showed that metric embeddings cluster into different groups, meaning the metric plays an important role as context. Although all clusters had similarly high average scores, the embeddings themselves still carry meaningful information.
- **Prompt–response similarity:** Cosine similarity between prompts and responses did not directly predict quality scores. Both high and low scores appeared across the full similarity range. This showed that the model needs more complex features to truly understand quality.

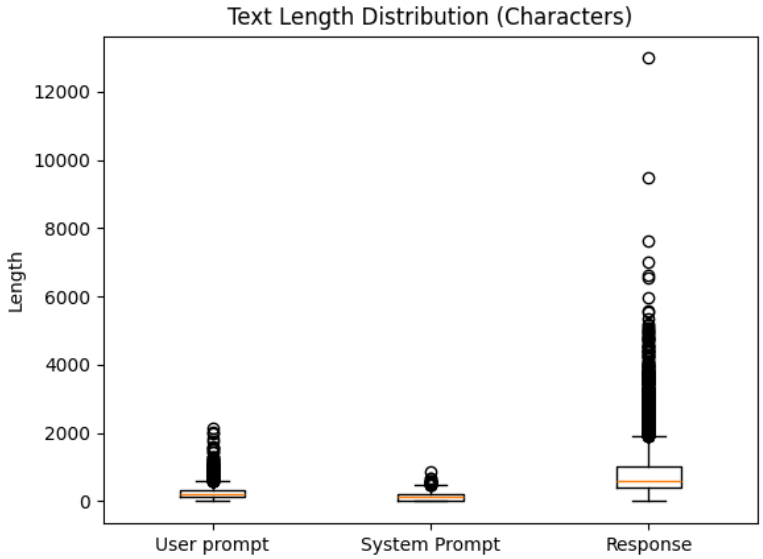

Figure 1: text length

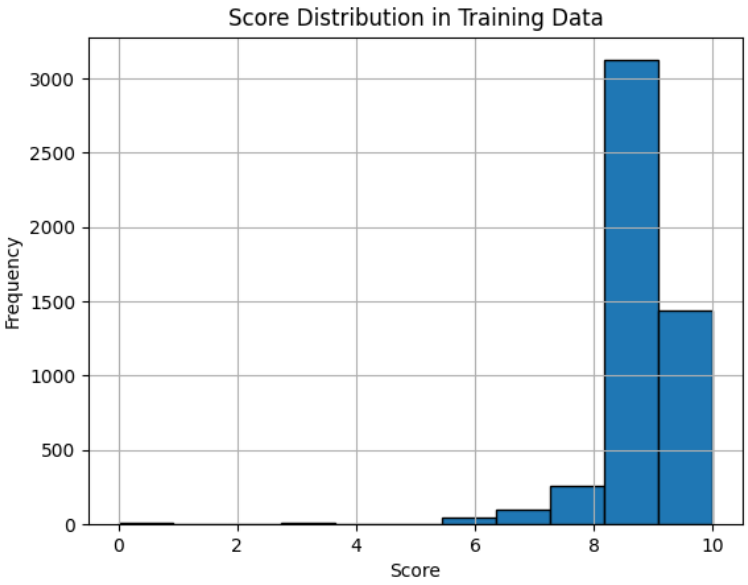

Figure 2: Score distribution of training data

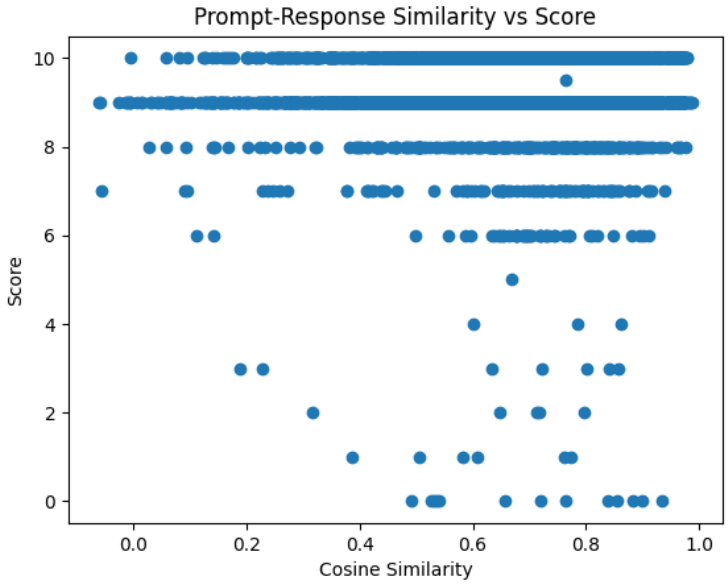

Figure 3: similarity between prompt and response

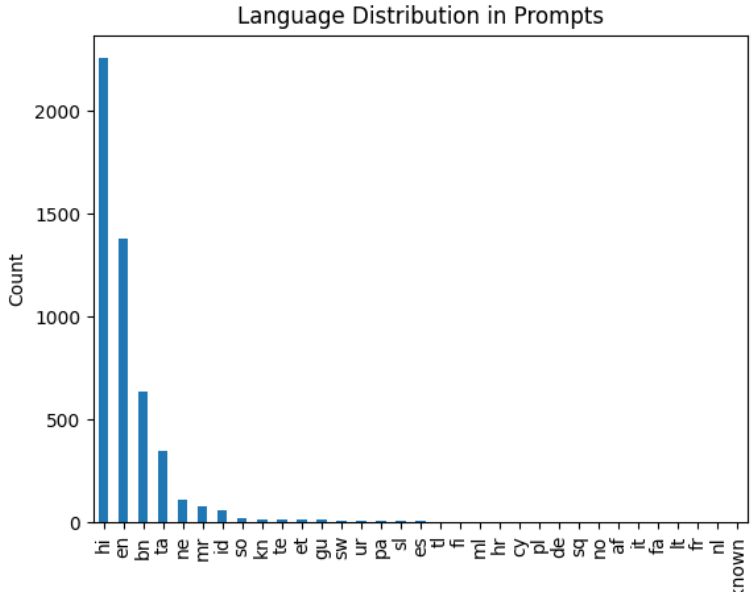

Figure 4: language distribution

Methodology and Pipeline Details

1. Configuration

The following configuration was used to maintain stable and reproducible code:

Parameter	Value	Motivation
AUGMENTATION_RATIO_NEGATIVE	0.5	Improve representation of low scores
AUGMENTATION_RATIO_PERFECT	0.5	Improve representation of perfect scores
N_SPLITS	5	Standard cross-validation
PCA_COMPONENTS_GLOBAL	30	Compress large embedding space
PCA_COMPONENTS_DIFF	15	Capture direction of deviation
PCA_COMPONENTS_PROD	15	Capture multiplicative interactions
np.random.seed	42	Reproducibility

2. Data Loading and Stratification

All four embedding matrices were loaded from NumPy arrays. Since scores are continuous, we created three coarse bins (low, medium, and high) to use in stratified K-fold validation. This ensured balanced splits.

3. Augmentation Strategies

Two augmentation strategies were applied:

Strategy	What It Does	Why
Synthetic Negatives	Creates mismatched embedding pairs with scores between 0 and 0.5	Helps model lea
Perfection Boosting	Copies perfect-score samples with noise	Helps stabilize

Only training folds were augmented. Validation folds remained untouched.

Advanced Feature Engineering

Feature Category	Description	Motivation
Raw Metric Name Embeddings	used pretrained embeddings of metric names directly	helps us capture the meaning behind metric names so the model understands what each goal really represents
Instruction Embeddings	embeddings of the system prompts using a Sentence Transformer	help describe the rules or guidance given, so the model can see what the expected behavior or output should be
Communication Embeddings	We combined user prompts and responses into embeddings using the same model	show exactly what was communicated and produced
Cosine Similarity (Metric vs Communication)	calculated similarity scores between normalized metric and communication embeddings	tells us how closely the response matches the intended goal that is captures semantic alignment
Element-wise Difference (Metric vs Communication)	took differences between metric and communication vectors	This highlights where responses differ from the goal, helping the model learn about specific mismatches
Element-wise Product (Metric vs Communication)	multiplied metric and communication vectors element-wise	understands the interactions and combines features, giving more insights on how goal and response relate
Cosine Similarity (Instruction vs Communication)	Similarity between instructions and communication was computed	measures how well the response follows the given instructions or rules
Element-wise Difference (Instruction vs Communication)	Differences between instruction and communication embeddings	to identify where the response deviates from what was asked or expected
Normalization	all vectors were normalised before calculating similarity or differences	Normalising ensures the comparisons are fair

These features were created to help the model better understand the relationships between the goals, instructions, and the actual responses. By looking at things like similarity and differences from various angles, the model can pick up subtle details about how well the outputs match what was expected. Normalizing the embeddings just makes sure everything is on the same playing field, so comparisons are meaningful and consistent. This way i was hoping the model gets richer information to learn from thus improving its predictions.

Model Design, Training, and Evaluation

Model Selection: LightGBM Regressor

LightGBM was chosen for its strong performance on tabular data and its ability to learn non-linear interactions. A conservative set of hyperparameters was used:

- n_estimators = 6000
- learning_rate = 0.01
- num_leaves = 40

- feature_fraction = 0.6
- bagging_fraction = 0.7

Training Strategy

- Stratified K-fold cross-validation
- Augmentation only applied to training folds
- Early stopping with patience of 200 rounds

Evaluation and Postprocessing

- RMSE was measured on cross-validation folds
- External leaderboard score reached **2.451**
- Predictions were clipped to [0, 10]
- Final outputs were rounded for submission

Alternative Method

An alternative approach I used was a 1D CNN. The raw vectors were combined into a single input matrix. Then the scores were log-transformed to stabilize variance.Batching was done and the raw scores were used to make a weighted MAE loss function that emphasized lower score samples.

The CNN architecture consisted of many convolutional layers with batch normalization, ReLU activations, and max pooling to extract features from the input sequence. After this global max pooling and a regression to predict the log-scaled quality score was used. Training was performed using Adam optimization with early stopping based on validation RMSE measured in the original scale. A 5-fold cross-validation was used.

This method was more time-consuming to train and tune.In comparison, fine-tuning the LightGBM on the engineered features was more efficient and achieved better predictive performance in this task.

Model Pipeline Flowchart