DA5401 Data Challenge Report: AI Evaluation Score Prediction

Saranath P
Roll No: DA25E003

November 21, 2025

1 Introduction

We know that Evaluating AI models by hand is slow and expensive.Therefore, In this challenge, we had to build a model that could look at a given User Prompt, a System Prompt, a corresponding AI Response, and a corresponding Metric Definition, and finally assign a score from 0 to 10.

The main challenges I faced were:

- **Multilingual Data:** The text wasn't just in English. It was also that a significant portion of the text was in Hindi and Tamil, and these languages were not in equal ratio. Some language data points dominated there.
- **Imbalanced Scores:** Almost all examples in the training data were "good" answers (scores of 9 or 10) with a clear class imbalance against the "bad" answers (scores 0-5).
- **Abstract Metrics:** The model had to understand complex rules like "Rejection Rate" or "Fluency." While it is intuitive for humans, it is difficult for models to grasp these concepts

2 Exploratory Data Analysis (EDA)

Before experimenting with any models, I analyzed the data to understand what I was dealing with.

2.1 Score Distribution

The first thing I noticed was a massive imbalance as highlighted above As seen in Figure 1, Most of the data has a score of 9 or 10. There were very few examples of "bad" responses (scores 0-5).

Figure 1: Distribution of Fitness Scores. Over 90% of the data is in the 8-10 range.

This was a major problem. If I trained a model on this raw data, it would likely just guess "10" for everything and get a high accuracy without actually learning anything. This obviously defeats the purpose.

I then performed a more detailed analysis (Figure 2) to look at various existing correlations. I found that the length of the response did not correlate with the score. This means the model needed to understand the *meaning*, not just count words.

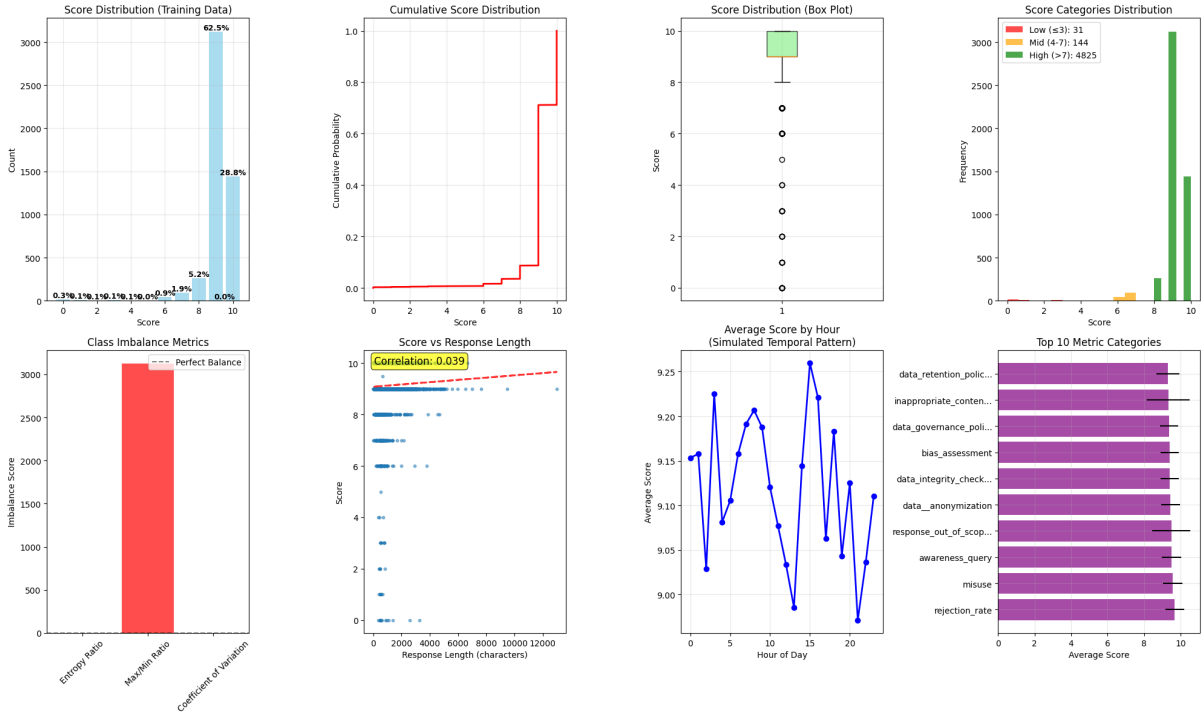

Figure 2: Detailed analysis showing the extreme skew and lack of correlation with response length.

2.2 Language Diversity

Using a language detection tool, I then found that the dataset was quite diverse. While English was dominant, there were significant amounts of Hindi and Tamil.

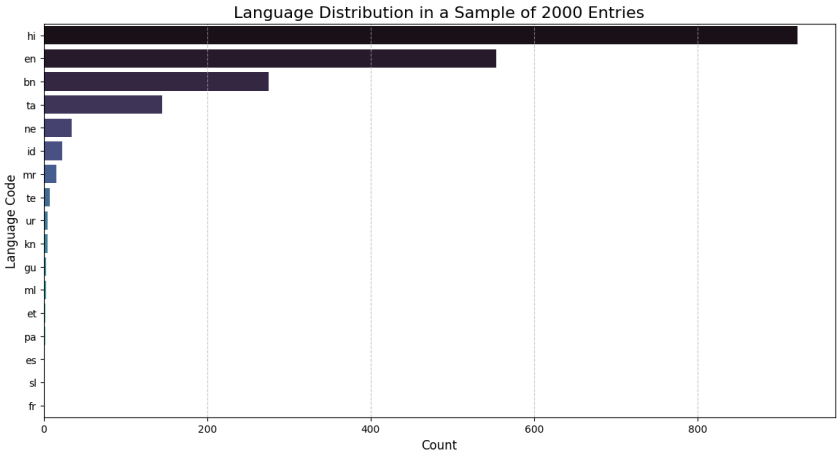

Figure 3: Language distribution showing a mix of English and Indian languages.

This insight helped me a lot. It meant I could not use standard English-only models like BERT. I needed a model that could properly understand Indian languages as well.

2.3 Metric Analysis

I also looked at specific metrics. For example, the "Rejection Rate" metric (Figure 4) is almost binary like it's either a perfect score or not.

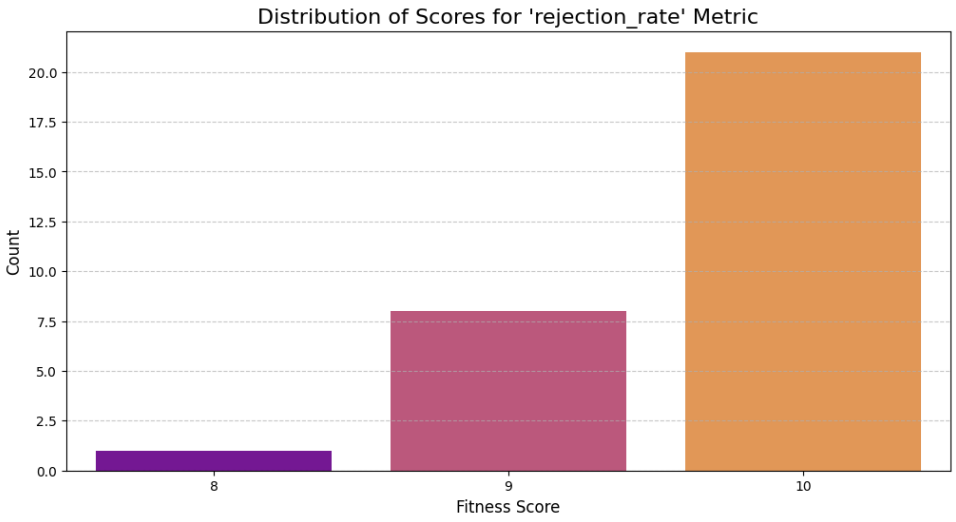

Figure 4: Score distribution for the 'Rejection Rate' metric.

3 Methodology

3.1 Initial Failed Approach: GRAML

At the start, I tried to implement a sophisticated method called "Goal Recognition as Metric Learning" (GRAML). This involved a Siamese Neural Network with LSTM encoders. I initially thought a complex architecture would capture the deep relationships in the text.

Why it failed:

1. **Too Complex:** I observed The model was too big for the small dataset (5,000 samples).
2. **Overfitting:** It memorized the training data but performed terribly on the test set.
3. **Slow:** It took a long time to train and tune.

3.2 Final Approach that worked best: Embeddings + LightGBM

I endedup gravitating to a more practical approach that focused on fixing the data issues rather than building a complex network.

3.2.1 1. Multilingual Embeddings (LaBSE)

To handle the Hindi and Tamil text found in the EDA, I used the `sentence-transformers/LaBSE` model. This model converts text into numbers (embeddings) and is designed to ensure that a sentence in Hindi and its translation in English have very similar numbers.

3.2.2 2. Data Augmentation (Negative Sampling)

Since I didn't have enough low scores (0-4), I created synthetic data. I took existing high-scoring pairs (Score ≥ 8) and swapped their metric definition with a random one.

- *Example:* Whaty I did was take a perfect response for "Grammar" and label it as "Safety."
- *Result:* The response is now a "bad" match for the metric.
- I assigned these new pairs random low scores (0-3).

This doubled my dataset size and taught the model what a mismatch looks like.

3.2.3 3. Feature Engineering

Instead of just giving the raw text to the model I calculated cosine similarities:

- Similarity between User Prompt and Metric.
- Similarity between AI Response and Metric.

This helped the model explicitly see how well the answer aligned with the rules.

3.2.4 4. LightGBM Regressor

I used LightGBM as the final predictor. It is fast, handles large feature sets well, and is generally more robust than deep neural networks on tabular data like this.

4 Results

The final model was trained with the following parameters:

- **Learning Rate:** 0.07
- **Num Leaves:** 150
- **Depth:** 7

Performance:

- Training RMSE: $\sim$0.75
- Validation RMSE: $\sim$2.56

While the gap between training and validation errors highlighted some variance, this was significantly better than the previously GRAML approach. The predictions on the test set followed the expected distribution.

5 Conclusion

This assignment taught me that understanding the data is more important than using big or fancy model.

1. The EDA showed me that I needed multilingual support, leading to the choice of LaBSE.
2. The score distribution showed me I needed more negative data, leading to the augmentation strategy.
3. The failure of GRAML showed me that simple well thoughtof features (like cosine similarity) often beat complex neural networks esp on small datasets.

The final solution yieldede us reasonably good results. I learnt a lot from this assignment and look forward to applying these lessons in future projects.