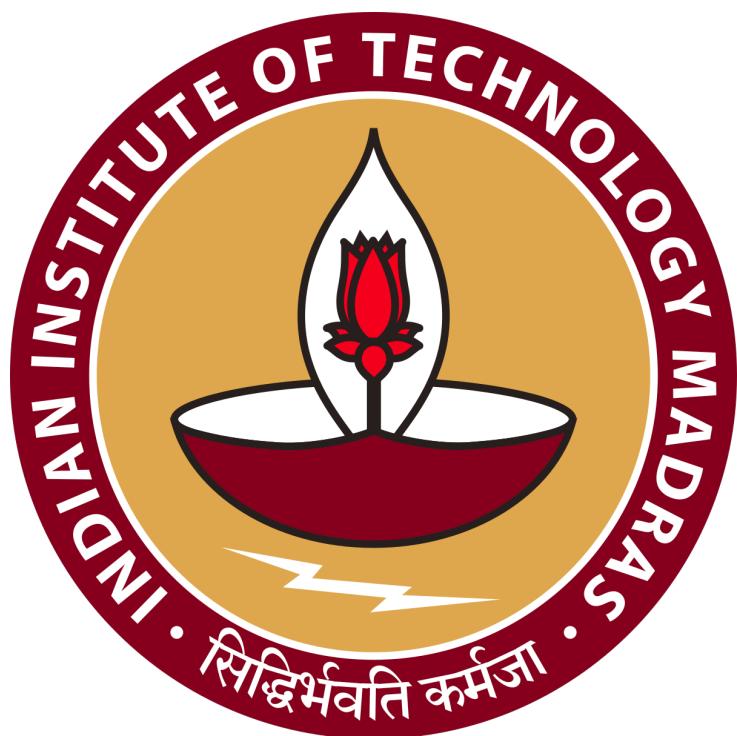


DA5401: Data Analytics Lab

End-Semester Report



Name: Yash Purswani

Roll No.: ME22B214

November 21, 2025

Contents

1	Problem Statement	2
2	Exploratory Data Analysis (EDA)	2
2.1	Data Overview	2
2.2	Data Augmentation	2
2.2.1	Random Negative Sampling	2
2.2.2	Semi-Hard Negative Sampling	3
2.2.3	Noise Injection	3
2.3	Text Processing	3
2.4	Feature Engineering	4
3	Methodology	4
3.1	Classifier	5
3.1.1	Model Selection	6
3.2	Expert Regressor	6
3.2.1	Model Selection	6
4	Implementation Tweaks	7
5	Results and Conclusions	7

1 Problem Statement

The end-semester Kaggle Data Challenge is based on metric learning, a type of machine learning that focuses on learning a distance function to measure how similar or different objects are from each other. In this competition, participants were asked to predict a similarity score between specific AI evaluation metric definition and a corresponding prompt-response text pair.

The goal of the challenge is to predict a score (0-10) for a given metric embedding and a corresponding prompt-response pair.

2 Exploratory Data Analysis (EDA)

2.1 Data Overview

The following insights were made by analysing and visualising the training data.

- The data originally had 5000 samples with 4 attributes, `system_prompt`, `user_prompt`, `response`, and `score` (0-10).
- There were multiple Indian languages like Hindi, Tamil, English, Sindhi, Oriya, Bengali, Assamese, etc. in the dataset.
- It is observed that some samples included multiple languages and even transliterated into other language scripts.
- The mean of scores is 9.1195 with a standard deviation of 0.9423.
- The original data is highly skewed towards high scores and mediocre scores had little to no representation.

2.2 Data Augmentation

Multiple techniques for data augmentation were used in order to account for skewness in original data.

2.2.1 Random Negative Sampling

For given sample, while keeping the prompt-response same, metric is randomly chosen from the list of given metrics and a very low score (0-2) is given.

Pros: Can quickly generate lot of augmented points and remove skewness.

Cons: The scores might not be accurate as there might be some similarity between metrics.

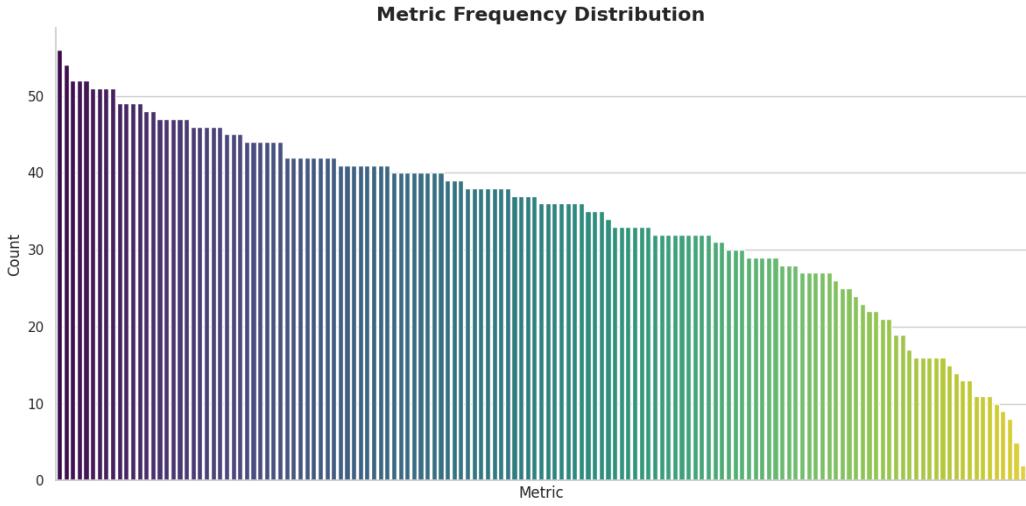


Figure 1: Metric Distribution in original training data

2.2.2 Semi-Hard Negative Sampling

Find metric and prompt-response pairs with medium cosine similarity to produce augmented samples with mediocre scores (4-6).

Pros: Increase the number of points in medium range.

Cons: May lead to overfitting this range.

2.2.3 Noise Injection

For samples with high scores, noisy samples can be generated by tweaking the responses, such as removing and adding some words, and reducing the original score.

Pros: Keeps the data realistic and prevents overfitting.

Cons: More noise can distort the semantic meaning.

2.3 Text Processing

The text string (`system_prompt`, `user_prompt`, `response`) in each sample is converted to embedding using `sentence-transformers` for training.

```
model =
    SentenceTransformer("l3cube-pune/indic-sentence-similarity-sbert",
    device="cuda")
x, y = [], []
```

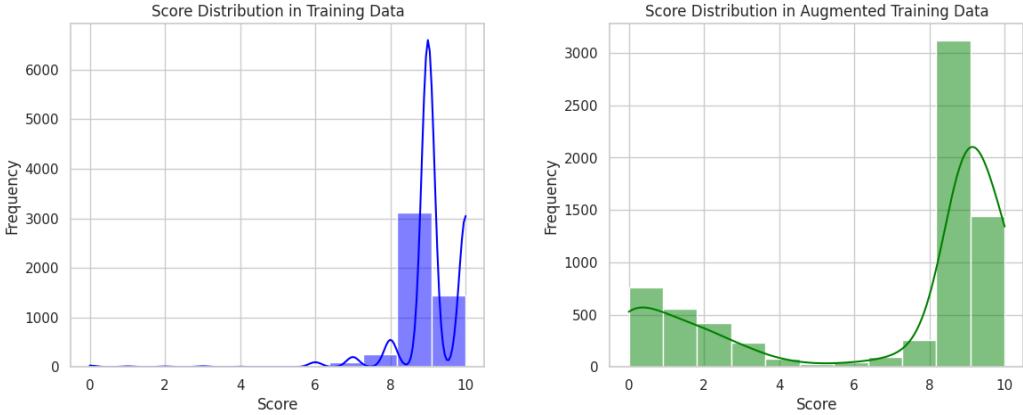


Figure 2: Left: Original distribution is left-skewed. Right: Augmented distribution

```

for r in tqdm(train):
    original_response = r['response']
    txt_good = f"r['system_prompt']}{[SEP]}{r['user_prompt']}{[SEP]}{original_response}"
    text_emb_good = model.encode(txt_good,
        normalize_embeddings=True)
    metric_emb =
        metric_embs[metric_map.index(r['metric_name'])]

    X.append(np.concatenate([text_emb_good, metric_emb]))
    y.append(r['score'])

```

13cube-pune/indic-sentence-similarity-sbert is chosen as it is currently the best performing transformer on Hugging Face for Indic languages.

2.4 Feature Engineering

Along with the text embeddings, some additional features were introduced to better capture the semantic relations.

3 Methodology

The general approach for going forward with this problem is **Mixture of Experts (MoE)** described as follows:

Feature Category	Features	Purpose
Length Features	len_user, len_resp, len_sys, len_text	Capture response verbosity
Token Counts	tok_user, tok_resp, tok_sys	Word-level granularity
Ratios	ratio_len_resp_user, ratio_tok_resp_user	Response-to-prompt proportionality
Overlap Features	cosine_similarity, overlap_count, overlap_frac_metric, overlap_frac_resp, overlap_jaccard, has_metric_in_resp	Metric-response alignment indicators

Table 1: Feature engineering categories, features, and their purposes.

1. The training data is divided into 2 categories based on low and high scores.
2. A classifier is trained to segregate samples into these two categories.
3. Two separate expert regressors are trained on the two categories to accurately predict the scores.
4. The final prediction is ensemble of the 2 experts weighted by its classification probability.

Let's look at the details of the approach.

3.1 Classifier

The training data is categorised into 2 classes based on the scores.

- **Class 0:** score range 0-5
- **Class 1:** score range 6-10

Classifier	F1-score	Pros	Cons
Logistic Regression	Re- 0.67	Very fast to train. Baseline model	Performs poorly on nonlinear relationships
Random Forest	For- 0.85	Captures non-linearity well	Becomes heavy on large dataset
LightGBM Classifier	0.95	Best performance for complex data	Sensitive to hyperparameters

Table 2: Comparison of classifiers

3.1.1 Model Selection

LightGBM Classifier is selected for its fast and robust performance in such a complex dataset. By performing a grid search using GridSearchCV the following hyperparameters were obtained.

```
clf_lgb = lgb.LGBMClassifier(
    objective="binary",
    num_class=2,
    n_estimators=500,
    num_leaves=31,
    learning_rate=0.05,
    class_weight="balanced",
    subsample=0.8,
    colsample_bytree=0.9,
    device='gpu',
    boosting_type='dart'
)
```

3.2 Expert Regressor

An expert regressor is trained on each class to give an accurate score prediction. The following models were tested for this task.

3.2.1 Model Selection

XGBoost Regressor is selected for its ability for capture such complex patterns in embedding space. Additionally its supports GPU acceleration for faster training and tuning. By performing a grid search using GridSearchCV the following hyperparameters were obtained.

Regressor	RMSE	Pros	Cons
Linear Regression	4.64	Extremely fast and simple	Underfits high-dimensional embeddings
Kernel Ridge Regression	3.05	Captures non-linear patterns	Slow for large datasets
LightGBM	2.31	Excellent performance on embeddings	Sensitive to hyperparameters
XGBoost	2.26	Robust to noise and outliers	Risk of overfitting without tuning

Table 3: Comparison of regression models

```
reg = xgb.XGBRegressor(
    objective="regression",
    n_estimators=250,
    learning_rate=0.03,
    num_leaves=64,
    subsample=0.9,
    colsample_bytree=0.8,
    device='gpu',
)
```

Predicted score using MoE model is given by

$$\text{pred}_{\text{MoE}} = p_{\text{class } 0} \cdot \text{reg}_{\text{low}} + p_{\text{class } 1} \cdot \text{reg}_{\text{high}}.$$

4 Implementation Tweaks

To further ensure robust performance and prevent overfitting a LightGBM Regressor is trained on the entire training dataset. An ensemble model of MoE and LightGBM Regressor is used as predicted score. This predicted score is further calibrated using `IsotonicRegression` to remove any misalignments from actual scores.

5 Results and Conclusions

- Data augmentation was important in improving coverage of medium and low-score areas.

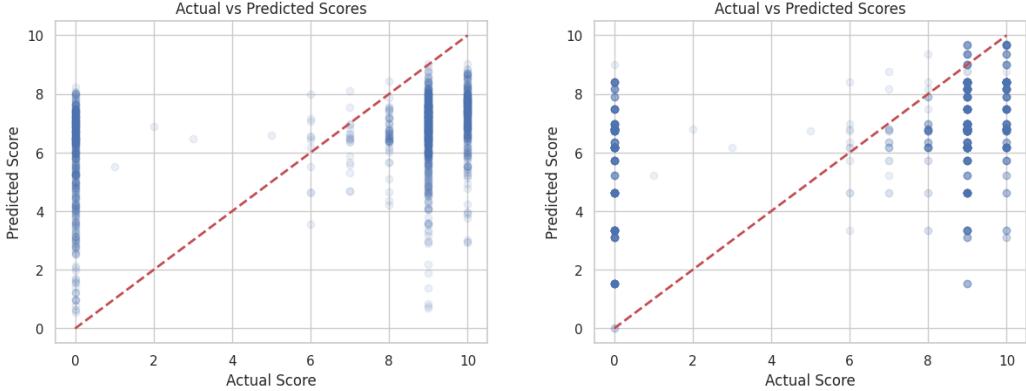


Figure 3: Left: Predictions from MoE. Right: Predictions after Isotonic Calibration

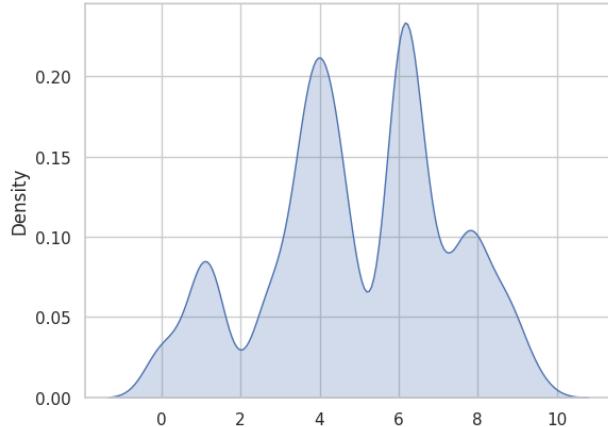


Figure 4: Score distribution on test data

- The Mixture-of-Experts (MoE) framework effectively handled the highly skewed scoring pattern. Each expert learned more specialized relationships.
- Isotonic calibration clearly improved the match between predicted and actual scores, as shown in the post-calibration scatter 3.
- The MoE approach, along with XGBoost regressors and a LightGBM classifier, achieved a test RMSE of 2.261.
- Overall, the final ensemble model (MoE + LightGBM + calibration) provided a balanced and reliable solution for similarity-score prediction.