Metric Learning for AI Evaluation: Multi-Stage Ensemble Approach

Your Name
Course: DA5401-2025

November 19, 2025

Abstract

This report describes my solution to the DA-Lab Challenge on predicting fitness scores between AI evaluation metrics and prompt-response pairs. I built a pipeline combining TF-IDF vectorization, SVD dimensionality reduction, geometric feature engineering, and ensemble learning. My approach achieved an RMSE of 2.096 on the test set. The key innovations include margin-based features from contrastive learning, synthetic negative sampling to handle class imbalance, and a stacked ensemble with metric-specific bias correction.

Contents

1 Introduction

The DA-Lab Challenge focuses on metric learning for conversational AI evaluation. Given a metric definition (as a 768-d embedding) and a prompt-response text pair, I need to predict their fitness score from 0-10. This measures how well the test case aligns with what the metric is trying to evaluate.

The dataset has 5,000 training samples across 145 metrics in multiple languages (Tamil, Hindi, Assamese, Bengali, Bodo, Sindhi, English). The main challenges are:

- Highly imbalanced scores skewed toward 8-10
- Multilingual text requiring robust feature extraction
- Abstract metric embeddings without actual definitions
- Potential train-test distribution shift

My solution uses geometric feature engineering to convert semantic similarity into distance metrics, augments data with synthetic negatives, and combines multiple models through stacking.

2 Data Preprocessing

2.1 Text Concatenation

I combine prompt, system prompt, and response into one text string separated by [SEP] tokens. This captures the full conversation context:

$$\text{text} = \text{prompt} \oplus [\text{SEP}] \oplus \text{system\_prompt} \oplus [\text{SEP}] \oplus \text{response}$$

2.2 TF-IDF Vectorization

I use TF-IDF with 10,000 features and trigrams (1,3) to capture technical phrases common in evaluation metrics. Sublinear TF scaling reduces the impact of very frequent words.

2.3 SVD Reduction

Both TF-IDF vectors (10,000-d) and metric embeddings (768-d) are reduced to 64 dimensions using Truncated SVD. This creates a shared space where I can compute meaningful distances between text and metrics.

3 Feature Engineering

Instead of using raw embeddings, I engineer 24 geometric features that capture relationships between text vector $\mathbf{t}$ and metric vector $\mathbf{m}$:

Distance Features (8): Cosine similarity, dot product, L1 and L2 distances, plus element-wise statistics:

$$\cos(\mathbf{t}, \mathbf{m}) = \frac{\mathbf{t} \cdot \mathbf{m}}{\|\mathbf{t}\|\|\mathbf{m}\|}$$

Prototype Features (8): I compute a centroid for each metric using training samples, then calculate the same distance features to these prototypes.

Margin Features (3): The most important features. For each sample, I compute:

$$\text{margin} = \cos(\mathbf{t}, \mathbf{p}_{\text{correct}}) - \max_{j \neq \text{correct}} \cos(\mathbf{t}, \mathbf{p}_j)$$

This margin tells me how much closer the text is to the correct metric versus the best wrong metric. High margin means clear match, low margin means ambiguity.

Text Stats (3): Length ratios between prompt and response.

4 Data Augmentation

Since 80% of training scores are 8-10, models struggle to learn what poor matches look like. I generate synthetic negatives by:

1. Taking each training sample
2. Randomly assigning a wrong metric
3. Recomputing all geometric features with this wrong metric
4. Assigning a low score (0, 1, or 2)

This doubles the dataset to 10,000 samples and balances the score distribution.

5 Model Training

I use a 4-stage approach:

5.1 Stage 1: Augmented Random Forest

Train Random Forest (500 trees) on all 10,000 samples including synthetics. This model learns to detect mismatches.

5.2 Stage 2: Base Models with Cross-Validation

Train three models on original 5,000 samples using 5-fold GroupKFold (grouped by metric):

- **Ridge Regression** (alpha=1.0): Linear baseline
- **Random Forest** (500 trees): Non-linear interactions
- **LightGBM** (300 trees, lr=0.05): Gradient boosting

5.3 Stage 3: Stacking with NNLS

Combine all four models using Non-Negative Least Squares on out-of-fold predictions. Final weights were:

- Ridge: 0.164
- RF: 0.039
- LGBM: 0.000
- Augmented RF: 0.829

The augmented RF dominates because learning from negatives is crucial.

5.4 Stage 4: Bias Correction

Apply metric-specific corrections using Bayesian shrinkage:

$$\text{bias}_j = \frac{n_j(\bar{y}_j - \bar{\hat{y}}_j)}{n_j + 5}$$

This fixes systematic over/under-predictions for specific metrics while avoiding overfitting through the shrinkage term.

6 Results

6.1 Cross-Validation Performance

Table 1: Out-of-Fold RMSE

Model	**RMSE**
Ridge	0.942
Random Forest	0.973
LightGBM	0.993
Augmented RF	0.915
Stacked Ensemble	0.720
Final (with bias correction)	**0.599**

6.2 Test Performance

Final test RMSE: **2.096**

The gap between CV (0.599) and test (2.096) reflects the documented distribution shift between train and test sets.

6.3 Feature Importance

Top 5 features from Random Forest:

1. Cosine margin (18.3%)
2. Cosine to correct prototype (15.7%)
3. Direct cosine similarity (12.4%)
4. L2 distance (9.8%)
5. Dot product (8.6%)

Margin features being most important confirms that relative distances matter more than absolute similarities.

7 Discussion

7.1 What Worked

Margin features: Explicitly modeling "how much better does the correct metric fit than alternatives" was the key insight.

Synthetic negatives: Without augmentation, models couldn't learn low-score patterns due to extreme imbalance.

Model diversity: Combining linear, tree-based, and boosting models captured different patterns.

7.2 Limitations

The train-test distribution shift is significant. My CV performance doesn't predict test performance well. Having only embeddings instead of actual metric definitions limited feature engineering possibilities. The multilingual aspect wasn't fully exploited—language-specific features might help.

7.3 Alternative Approaches

I considered transformer fine-tuning (BERT/RoBERTa) but 5,000 samples seemed too small and training would be slow. Metric-specific models would overfit with only approximately 34 samples per metric. Neural metric learning (Siamese networks, triplet loss) could be promising future work.

8 Conclusion

I achieved competitive performance (RMSE 2.096) through careful geometric feature design, data augmentation, and ensemble methods. The key lessons are:

1. Contrastive margins effectively capture alignment
2. Synthetic negatives address class imbalance when real negatives are rare

3. Multi-stage ensembles outperform individual models

4. Metric-specific calibration improves heterogeneous predictions

Future improvements could include better embeddings (Sentence-BERT), meta-learning across metrics, uncertainty quantification, and active learning loops.