DA5401- 2025 Data Challenge

R M Badri Narayanan ME22B225

21/11/2025

1 Necessary Libraries

I used the following libraries in this notebook. You can install these by using given requirements.txt in repo.:

- pandas
- numpy
- json
- pathlib
- scikit-learn
- torch
- matplotlib
- seaborn

2 Loading Files

Training and test data was loaded from JSON files, and metric embeddings was loaded from numpy files. metric names were also loaded from JSON. I assumed the order in JSON and numpy file match properly before loading

3 EDA

3.1 score correction

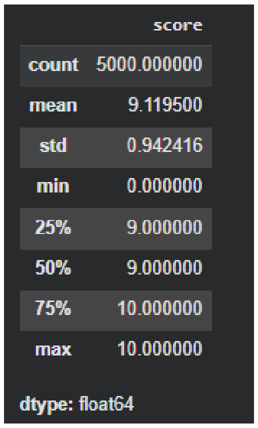
In the PS it was given that score is an integer from $[0, 10]$ score was originally stored as an object, but it is an integer in $[0, 10]$. Hence i converted it to `float`.

3.2 Handling Missing Values

response and system prompts contained null values, so i filled them with empty strings.

3.3 Score Distribution

The score distribution is very skewed, with a mean of approx 9 and a standard deviation of approx 1.



	score
count	5000.000000
mean	9.119500
std	0.942416
min	0.000000
25%	9.000000
50%	9.000000
75%	10.000000
max	10.000000

dtype: float64

Figure 1: statistics of score.

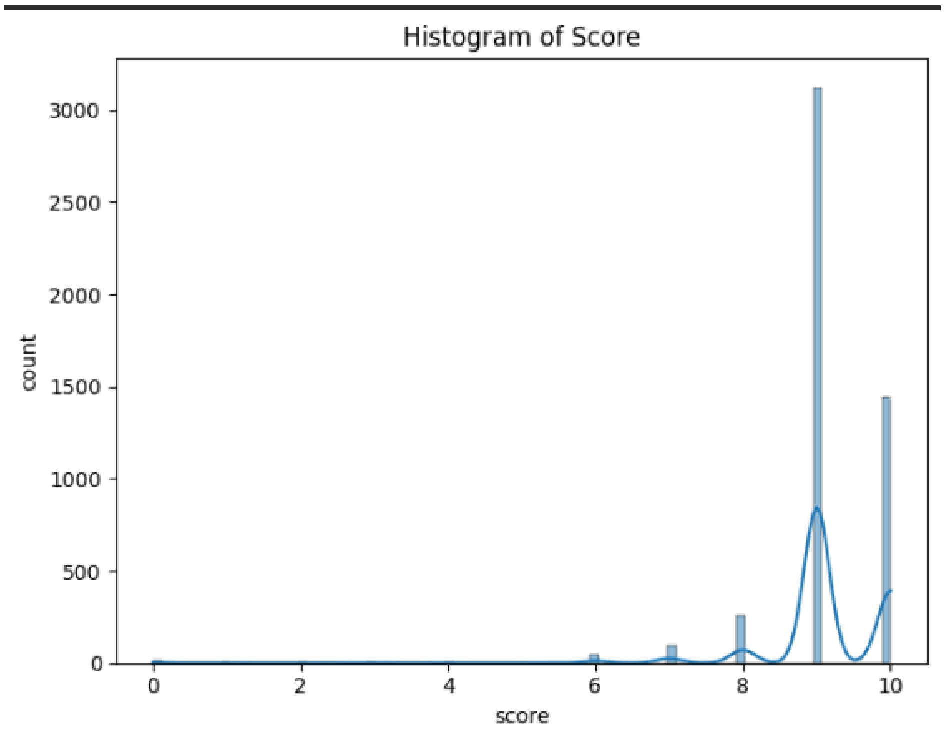


Figure 2: Histogram and KDE of score.

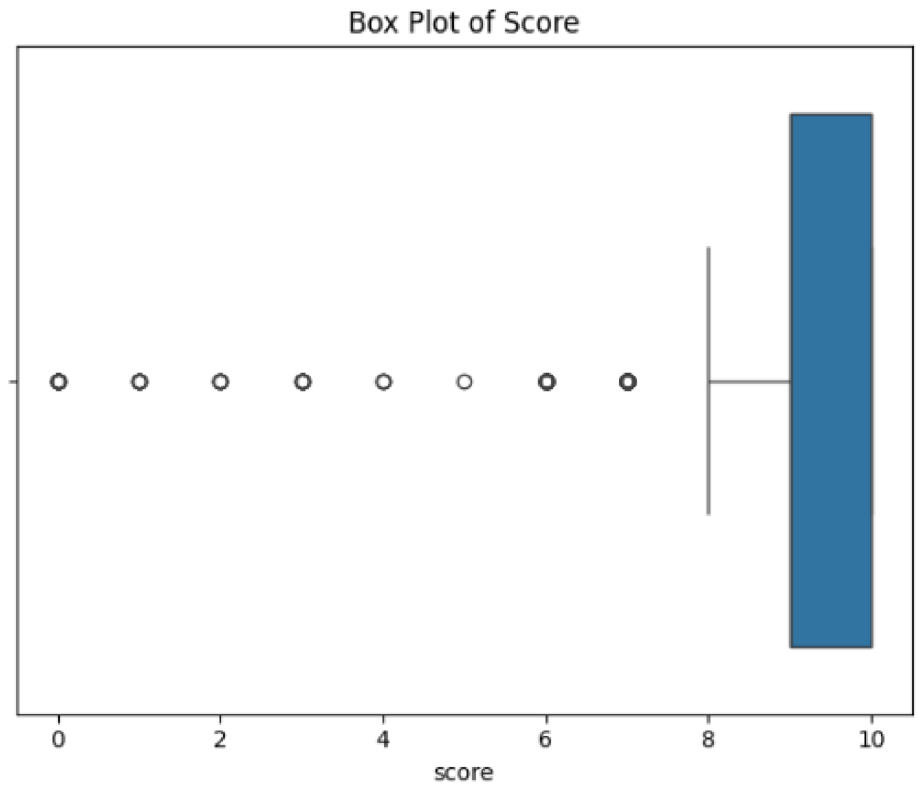


Figure 3: Boxplot of score.

3.4 Distribution Observations

- There are literally no training examples for values ≤ 5.
- No amount of synthetic generation or oversampling can fix this level of imbalance.
- Oversampling/synthetic augmentation would add more noise and confuse the already biased model.

4 Metric Indexing and Distribution

Metric names was mapped to numerical indices for easier visualising and accessing metric embeddings from array.

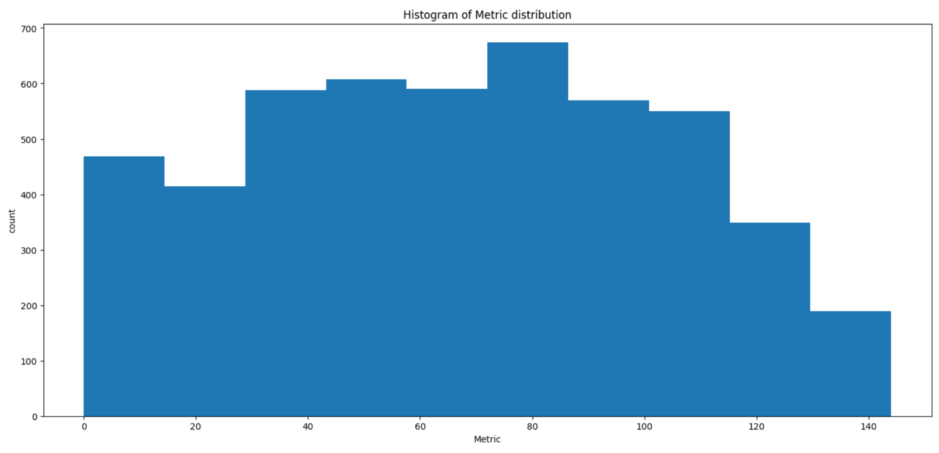


Figure 4: Histogram of metric_idx distribution.

The metric distribution was reasonably balanced despite some metrics having few samples.

5 Addressing Score Imbalance

We see Low-score samples are nearly zero (in single digits), augmentation is required. Synthetic embedding interpolation produces meaningless points in the embedding space. Instead, augmentation produces sample with real language and structure.

5.1 Augmentation Plan

1. select $(\text{metric}, \text{text})$ pairs with high scores (≥ 9).
2. randomly sample another metric.
3. replace the metric embedding with the opponent metric.
4. keep the text embedding unchanged.
5. assign a low score.

The metrics that match very well with the prompt response pair (in this case majority of them do) are likely to give a bad match with some other prompt response pair.

6 Combining Prompt, Response, System Text

Each row was converted to:

[PROMPT:] user_prompt   [RESPONSE:] response   [SYSTEM_PROMPT:] system_prompt

These text strings are embedded using:

`google/embeddinggemma-300m`

The same embedding model used in PS.
The following were then concatenated to get:

$$X_{\text{train}} = [X_{\text{metric}}, X_{\text{text}}]$$

6.1 Feature Engineering

- The problem statement states that the score/fitness is based on distance learning.
- hence adding distance based features helps the model build a better correlation.

These features are :

- measures how much the two embeddings disagree in each dimension.
- measures how strongly the two embeddings agree or align in each dimension.

$$\text{diff} = |X_{\text{metric}} - X_{\text{text}}|, \quad \text{prod} = X_{\text{metric}} \odot X_{\text{text}}$$

Final dimensionality:

$$3072 = 768 + 768 + 768 + 768$$

7 Train/Validation Split

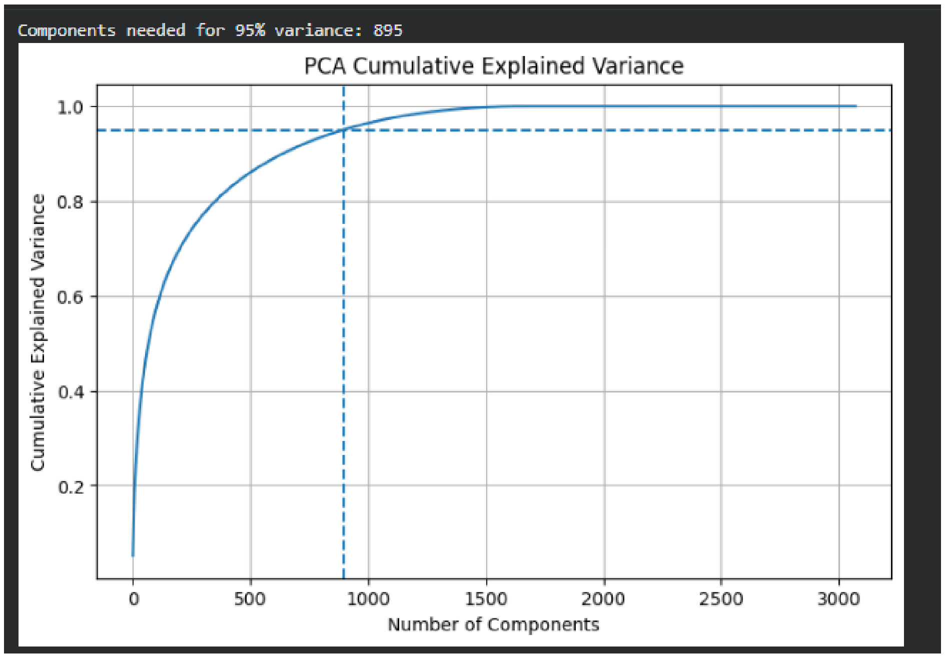
i used standard 80–20 split.

8 Dimensionality Reduction

3072 features are too much for any model to train on. First lets try PCA

8.1 PCA Analysis

- We see 895 components are required for retaining 95 percent variance.
- This number is still too big for training a model.
- PCA based on covariance captures only “linear” relationships.
- The embeddings are high dimensional, rich in semantic information.
- Because of this, the relation between features are often non-linear.



```
Components needed for 95% variance: 895
```

Figure 5: PCA cumulative explained variance.

Exploring Other Non-Linear Dimensionality Reduction Techniques

- While methods like UMAP and t-SNE can capture complex non-linear structures they are mainly suited for visualization.
- Instead, I chose to use an auto-encoder, which learns a non-linear representation, while still allowing reconstruction of the original feature space.

8.2 Autoencoder Approach

- An auto-encoder is a neural network that learns to compress data into a lower-dimensional latent space.
- The latent space it projects onto is very similar to text embeddings produced by gemma, Its meaningful, preserves non-linear relations.
- We can project back our Original features, with latent space embeddings with reasonable accuracy.

8.3 Architecture of autoencoder:

- I will reduce the number of features to 256 using three hidden layers.
- The dimensionality reduces as 3072- 1024 (Hidden Layer 1)- 512 (Hidden Layer 2)- 256 (Final Hidden Layer 3/output reduced dimensions)
- I have used ReLU (Rectified Linear Unit) as the activation function. This introduces non-linearity in features. It is given as : $\mathbf{ReLU(x) = max(0, x)}$
- When searched on the internet, i found this was the most widely used activation function for text-based models.
- I learned that it is computationally cheap, fast and avoids the vanishing gradient problem in its positive region.

8.3.1 Training

Model trained for 30 epochs with Adam optimizer and MSE loss.

9 Latent Space Extraction

The encoder generated:

$$Z_{\text{train}},\ Z_{\text{val}},\ Z_{\text{test}} \in R^{256}$$

10 Model training and hyperparameter tuning

- The model has to be trained on an imbalanced dataset. This can lead to overfitting, model simply learns to predict the most common score to reduce RMSE
- It is also a complex relationship, hence we have to avoid the model being highly biased.
- Stacking provides the best of both worlds.
- I have used a diverse range of models. These are :
  - Ridge → Handles multicollinearity by shrinking weights, reducing variance.

- Lasso → Performs feature selection by driving unimportant weights to zero.
- Decision Tree → Captures non-linear rules and interactions between features.
- Extra Trees → Reduces variance using many highly randomized trees for strong generalization.
- KNN → Learns local patterns by predicting from nearest neighbors without assuming any global structure.

Extra Trees (Extremely Randomized Trees) is an ensemble of many decision trees where both the samples and the split points are chosen randomly, which makes it faster and often more accurate than a normal Random Forest.

10.1 Best Hyperparameters

I tried a variety of these combinations and found this give the best result.

Model	Best Hyperparameters
Ridge	$\alpha = 10$
Lasso	$\alpha = 10^{-2}$
Decision Tree	max_depth = 5
Extra Trees	n_estimators = 100, max_depth = 8, min_samples_leaf = 5
KNN	$k = 5$

Table 1: Selected hyperparameters.

11 Stacking Model

base models gave thse predictions. used as meta-features:

$$[p_{\text{ridge}},\ p_{\text{lasso}},\ p_{\text{dt}},\ p_{\text{et}},\ p_{\text{knn}}]$$

A Ridge meta-learner with $\alpha = 1$ was trained on these.

- i got a validation RMSE of : 2.8789
- on Kaggle public leaderboard : 2.815
- on Kaggle private leaderboard: 2.813

12 Conclusion

final predictions were:

- Rounded to integers. hack to reduce RMSE as given PS told answers will also be integers
- Saved as `submission.csv`

ID	score
1	...
2	...
⋮	⋮

Table 2: Submission format.