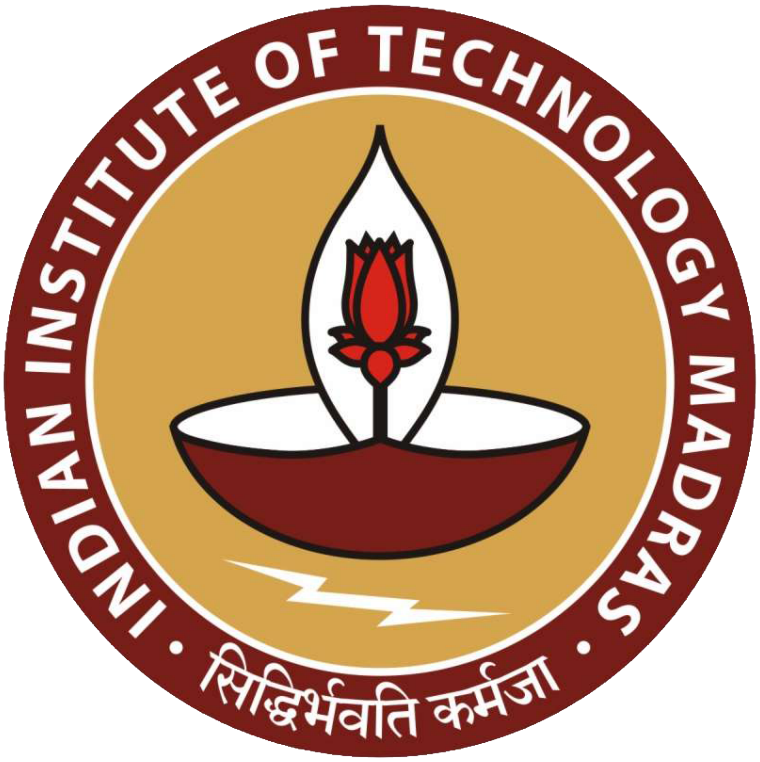
Department of Aerospace Engineering
Indian Institute of Technology Madras



Data Analytics Lab

Hackathon Report

U Shriram
AE22B008

1 Introduction and Problem Formulation

The objective of this challenge was to build a regression model capable of predicting the score (on a 0.0 to 10.0 scale) of LLM-generated responses based on word embeddings.

Our initial feature space comprised 1536 dimensions, got from the concatenation of embeddings of metric name meanings and combination of prompt/system prompt and response.

Main Problem: Severely skewed dataset with majority of the scores being 9.0 and 10.0.

The model used to convert the text prompts into embeddings was Google's **MuRIL** model.

I used the MuRIL (Multilingual Representation for Indian Languages) embedding model, developed by Google because it was specifically trained on Indian languages.

2 Initial Approach: Classification

Initially I chose to handle this problem as a classification problem because of the large skew.

2.1 Discretization and Prediction Method

1. **Binning:** The 0.0–10.0 score was discretized into 5 bins (e.g., [0-6], [7], [8], [9], [10]). Thus I converted the target variable into distinct classes suitable for classification training.

2. **Training Objective:** A classifier (Random Forest, XGBoost and Support Vector Classifier) was trained using techniques like `class_weight='balanced'` and i also tried upsampling lower score bins using SMOTE to equally penalize misclassification across all five bins, thereby forcing the model to learn the patterns of the minority low-score classes.

3. **Score Prediction:** To restore the continuous nature of the score, the classifier's output probabilities were converted back into a final score using a weighted average of the bin values (S_{avg}):

$$S_{avg} = \sum_{k=0}^{N} P(\text{Bin}_k) \cdot \overline{Score}_k$$

   where $P(\text{Bin}_k)$ is the predicted probability for bin k, and $\overline{Score}_k$ is the average true score of training samples within that bin.

2.2 Flaw of the Classification Hypothesis and some results

The loss in classification treated the error between bin 1 and bin 5 same as bin 4 and bin 5 but ideally bin 4 and 5 are way more closer than bin 1 and 5.

Hence the loss function was the fundamental flaw in the problem formulation.

The results for some of the submissions can be seen in Figure 1. The best performing classification model was random forest getting the best rmse score of 4.265.

Considering nothing else was to be done in classification i moved onto regression models.

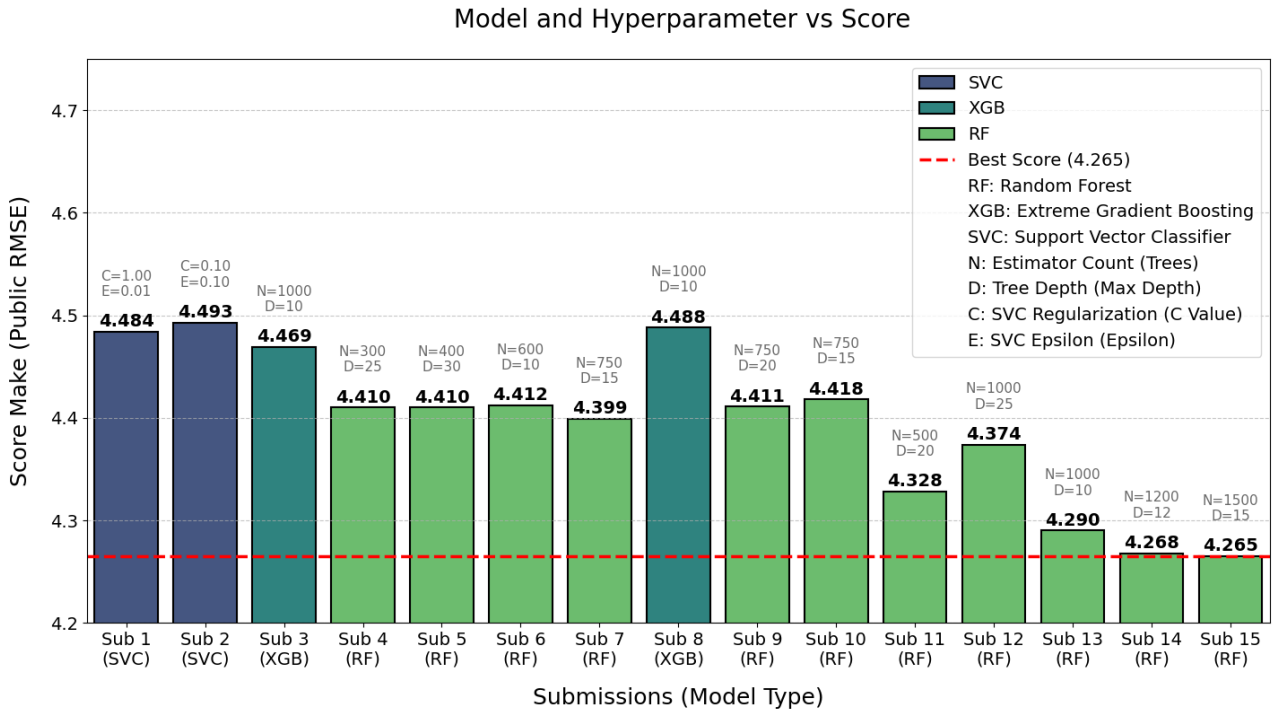


Figure 1: Model and Hyperparameters vs Scores

3 Regression: ensemble models

I performed feature engineering such as cosine similarities between different embeddings and added it to the training data.

I proceeded to model an ensemble of diverse regression models: Random Forest Regressor, XGBoost Regressor , and Support Vector Regressor (SVR).

These three models was picked because SVR is robust to noise, Random Forest is a bagging model which make a low bias high variance model better and XGBoost which makes a low variance high bias model better boosting prediction accuracy.

Finally a ridge regression was used to get the final output prediction.

1. **Binning for Weight Calculation:** The continuous target scores (y) were discretized into the same common bins (e.g., [0-6], [7], [8], [9], [10]).
2. **Weight Calculation:** Inverse class frequency weights were computed for these bins using the formula:
$$w_i = \frac{N_{\text{total}}}{N_k}$$
where N_{total} is the total number of samples and N_k is the count of samples in bin k.
3. **Training (Sample Weight):** Each regressor in the ensemble was trained using the `sample_weight` parameter:
$$\mathcal{L} = \sum_{i=1}^{N} w_i \cdot (\hat{y}_i - y_i)^2$$

We thus used an ensemble model along with weighted squared error for training our model.

3.1 Problem faced and some results

Because of very low samples score of 5 and 6. They had relatively high weights. So the model learnt the predicting 6 instead of 9 was cheaper than predicting 9 instead of 6 (an eg.).

Thus this model started predicting towards the mean of the dataset scores that is 5 6 and 7. I tried smoothening out the weights given to each bin to make it not so drastically different but that didn't quite work either.

Never the less it worked better than the classification models and some submission scores can be seen in Figure 2

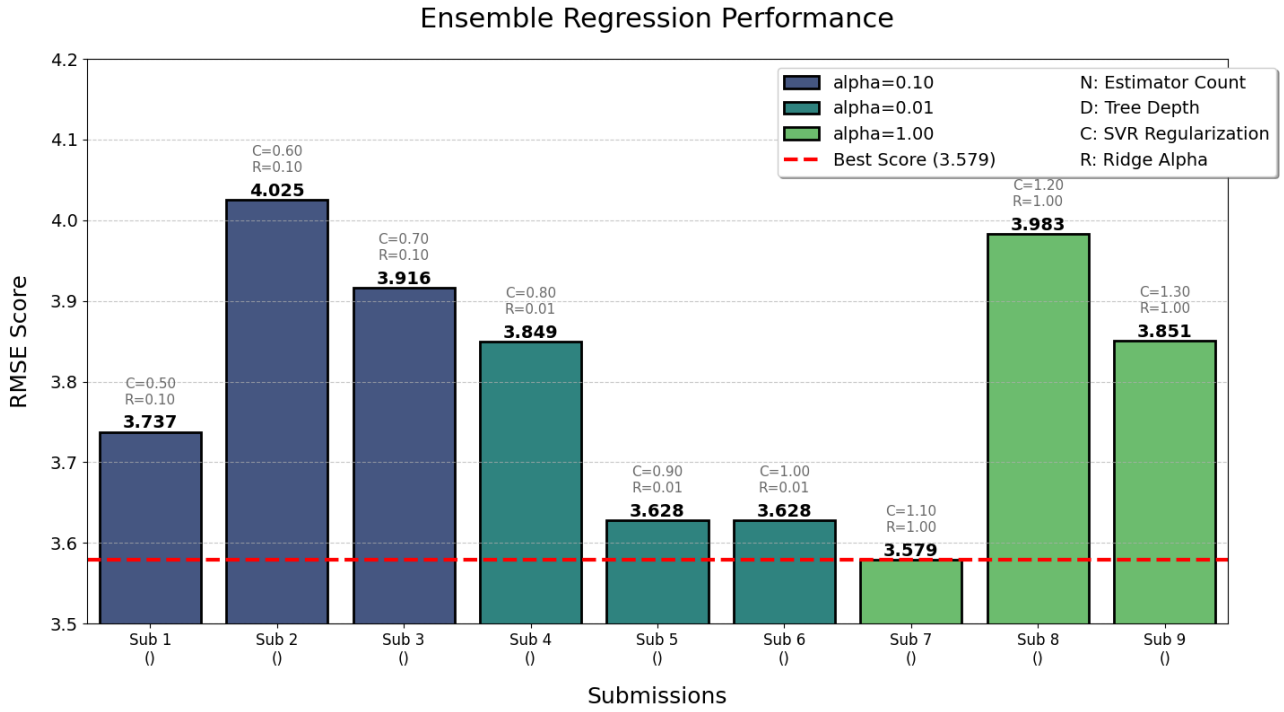


Figure 2: Ensemble model vs score

4 Negative Sampling, Regression with XGBoost

I generated synthetic data samples that teaches the model the difference between good and bad responses, a technique taken from contrastive learning which is called Negative Sampling.

1. **Type 1: Hard Negative Metric Swap**
   - When a specific metric meaning embedding has only high scores (≥ 8.0) in the dataset.
   - I assumed that the model assumes it doesn't know what a bad answer looks like for this metric.
   - A new feature vector is created by taking the current sample's Prompt/Response pair but we replace the original Metric meaning Embedding with a random Metric meaning Embedding.
   - I gave the score for this sample is as 0.0.
2. **Type 2: Contrastive Prompt Response (PR)-MixUp**
   - Applied when a metric meaning embedding has both high (≥ 5.0) and low (< 5.0) scores associated with it.

- I took a pair of vectors, a PR vector with high score and another with low score. A synthetic vector (PR_{Mix}) is generated by linear interpolation,

$$PR_{Mix} = \lambda \cdot PR_{Low} + (1 - \lambda) \cdot PR_{High}$$

  where λ is randomly chosen in the range [0.5, 0.9] to push the mix towards the bad sample.
- The score is also interpolated: $S_{Mix} = \lambda \cdot S_{Low} + (1 - \lambda) \cdot S_{High}$.
- This generated score more in the middle like 4 5 or 6 which now balances the high count of 9's and 10's and the high count of 0's generated from before method.

The final model was trained on the combined original and augmented dataset. I then trained an XGBoost regressor on this new dataset to obtain the results which an be seen in Figure 3

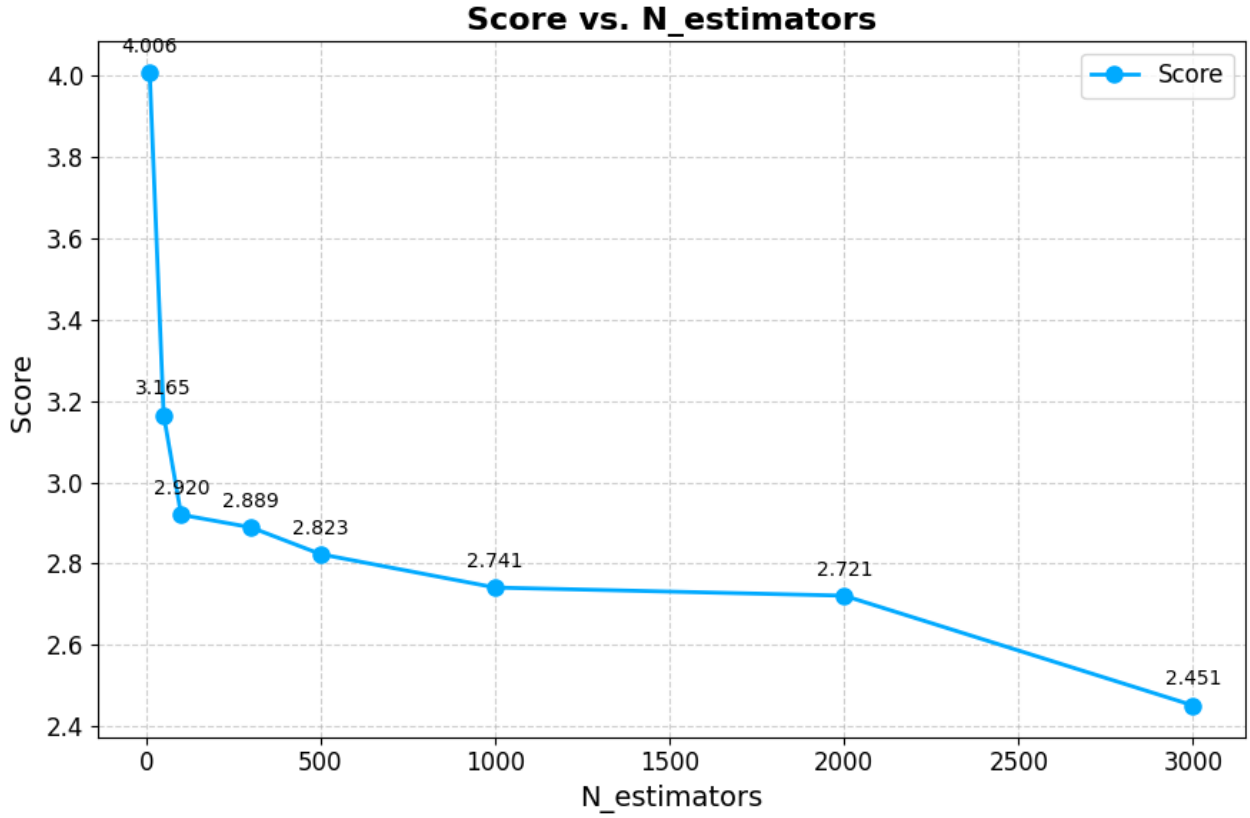


Figure 3: XGBoost + Negative Sampling vs score

It was seen that some changes to tree depth and some ways to change the dataset generation by tuning the number of synthetic samples changed the score but in general increasing value of n-estimators gave us a better rmse score. But it was computationally too time consuming after this point.

5 Conclusion

We finally found that negative sampling was the way to go about increasing the lower scores in order to find a way to teach our regression model a way to learn between good responses and bad ones.

NOTE: It is worth mentioning that this was few of the ideas I tried and what worked in the end from an extensive list of ideas tried.