DA5401 : End-semester Data Challenge Project Report

Aditya Tamras (BE22B007)

Introduction

The alignment of a model's response with a given evaluation metric helps in assessing conversational AI systems. This project aims to predict a fitness score (a value between 0 and 10) assigned by a large, sophisticated LLM judge. The dataset includes the elements of textual descriptions of evaluation metrics, prompt response pairs and corresponding fitness scores. The intended behaviour or the quality criterion (eg., fluency, relevance, safety) and the actual conversation instance to be assessed is reflected in each metric and prompt response pair respectively.

The task at hand includes training a model interpreting metric description as well as the conversational pair, encoding them and subsequently, predicting the extent of alignment of the response and the specified metric. Compatibility between the two pieces of text, i.e., semantic alignment, is implicit in the model, which highlights characteristics of metric learning. Since this setting also involves target outputs as continuous scores, it resembles regression. The project aims to record the fundamental relationship between an abstract metric and the evaluated concrete test case by generating embeddings for the metric and the conversation through the use of a SentenceTransformer model, constructing the interaction features and multiple regression architectures.

Part A : Data Loading and Preprocessing

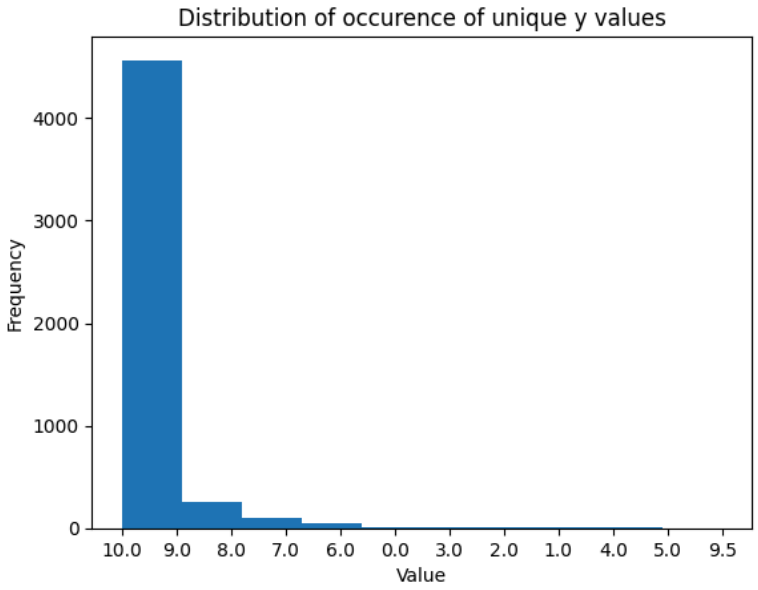
A SentenceTransformer model has been used to create dense vector embeddings for prompt/response text. The metric_embeddings.npy file is the direct source for the metric embeddings. For semantically similar texts to have closer vectors in Euclidean/cosine space, SentenceTransformer models align sentences and text pairs to fixed-size dense vectors. These models are based on transformer architectures such as BERT/DistilBERT. To induce sentence-level embeddings that are deemed appropriate for downstream supervised tasks, semantic similarity and retrieval, they are usually regulated to objectives such as Siamese/triplet losses. These embeddings function as metric text useful for regressors and compact, uninterrupted representations of prompt/response.

Empty strings have supplanted the None/NaN values noted in the data column of 'system_prompt'. This ensures that there are no errors in the regression algorithms and concatenation, tokenisation and embedding generation (text processing steps). The data structure is maintained by converting missing values to empty strings while adhering to the same input format across all samples, as most models do not function well with missing values.

Integrating the combined text and metric embeddings, and their linear combinations generate the features for the training of the data. The alignment or similarity between the two embeddings is captured through the inclusion of element wise interactions (absolute difference and product), which is otherwise not represented by simple concatenation.

Part B : Resampling with SMOTE

The scores corresponding to every prompt response pair, which is based on specific evaluation metrics, is stored in the labels (y).



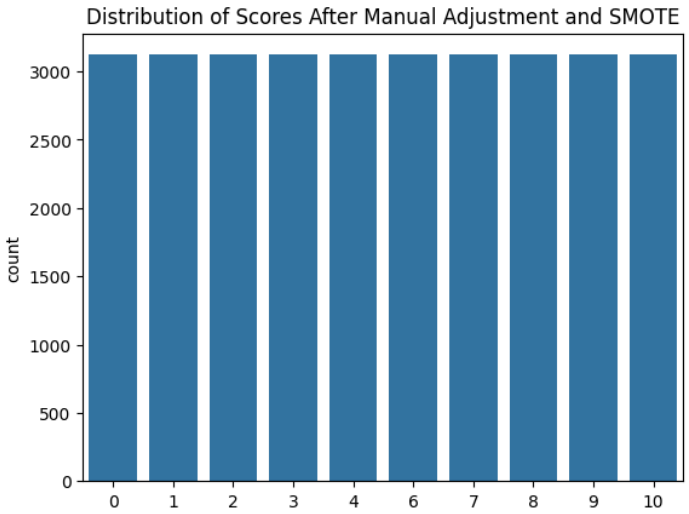
The above plot indicates that the data is highly skewed. The metric score for most of the training data points is 10. A regression model trained on skewed training data can overfit and lead to bad testing results. Hence, it is important to resample the data to make the distribution of scores more uniform.

How does SMOTE help?

By interposing between existing minority-class examples and their *k* nearest neighbours, synthetic samples are generated for minority classes via SMOTE. A neighbour is randomly selected for each minority sample and a new sample is created along the line segment between them to increase the effective sample size of the underrepresented classes without simply duplicating samples. SMOTE undertakes this to incorporate better balanced decision boundaries in regressors.

SMOTE's assumption is of the feature space being reasonably smooth for interpolation. However, noise can be introduced by synthetic points at times in high dimensional or highly nonlinear embedding spaces.

Classes for sampling were created after discretising/rounding *y*. SMOTE was applied after this step as the label distribution is skewed. The notebook handled classes with a simple example. To gain a better balancing training set, SMOTE was applied after perturbing that single label, to ensure no failure in SMOTE. Continuous regression targets were utilised for model training after the resampling (i.e., resampling was used to generate more training examples representative across score values).



The graph above demonstrates the distribution of labels after resampling. It is observed that the number of data points corresponding to every unique y label are similar and thus, the data distribution is more uniform.

Part C : Model Training

The model starts with simple Linear Regression and further builds up to the Boosting algorithm. The models are trained on original data and the data resampled with SMOTE and a comparative analysis is done to choose the best possibility.

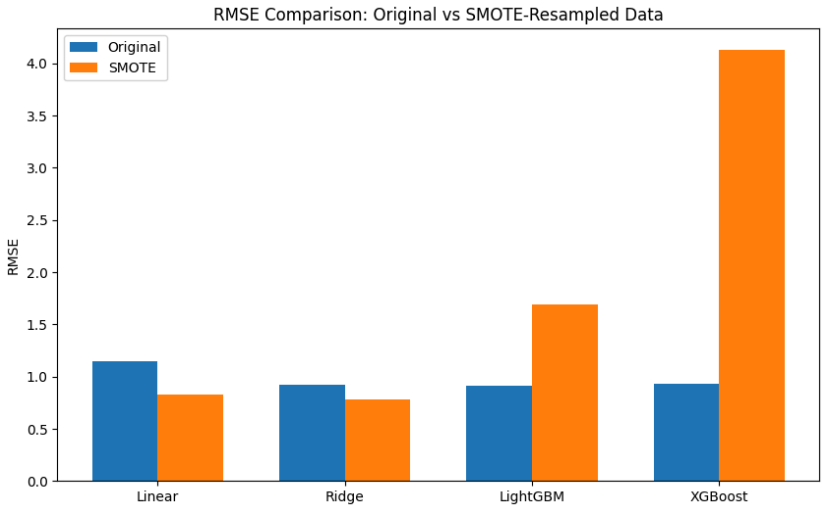
The implemented models include: Linear Regression, Ridge Regression, LightGBM and XG Boost Regression on the original and resampled data.

Summary of results:

Best model: Ridge Regression (selected based on submission/validation results).

Effect of SMOTE: SMOTE improved coverage for simple linear models but high RMSE was observed for more complex tree-based models (LightGBM & XGBoost), suggesting synthetic samples were beneficial for linear decision surfaces but possibly introduced noise for nonlinear

learners working in high-dimensional embedding spaces.



The above plot shows the comparative analysis of the results obtained on the original data and the data resampled using SMOTE

Deployment: Ridge predictions on test features were rounded and exported to da5401_submssion.csv.

Conclusion and Future Work

This project built a practical pipeline that transforms metric and text data into dense embeddings, constructs interaction features, addresses label imbalance with SMOTE, and evaluates a range of regression models. Empirically, Ridge regression produced the best validation/submission performance for this dataset and problem setup. The observations about SMOTE's differential impact on simple versus complex learners highlight the importance of matching resampling strategies to model inductive biases and feature geometry.

Some advanced strategies can be implemented in the future to enhance predictive performance. Some of the strategies are implementing ensemble and stacking techniques, where linear and non-linear models are combined to leverage complementary strengths. This can be complemented by metric shuffling,hyperparameter optimization (using grid search, randomized search, or Bayesian optimization) to more effectively tune complex models such as gradient-boosted trees, multi-fold cross-validation, refining the text embedding either by fine-tuning SentenceTransformer models or incorporating task-specific contrastive objective to yield richer semantic representations or applying dimensionality reduction methods such as PCA or learned bottleneck layers, which can help mitigate noise in high-dimensional embedding spaces and improve robustness across models.