Grammar Scoring Engine Report

# 1. Project Overview

The goal of this project is to develop a Grammar Scoring Engine that predicts grammar quality scores from spoken audio samples. The solution extracts audio embeddings using HuBERT, followed by training a Support Vector Regression (SVR) model for prediction.

This report details each part of the pipeline, explains model choices, and highlights what has been tested and what failed.

# 2. Data Preprocessing and Embedding Extraction

Audio samples are stored as .wav files in `audios\_train` and `audios\_test`, while `train.csv` contains filenames and grammar scores. To use these for machine learning, we extract HuBERT embeddings using the Facebook `hubert-base-ls960` model from HuggingFace.

HuBERT (Hidden Unit BERT) is chosen because it provides strong performance on audio representation tasks and is computationally lighter than Wav2Vec2. We use the mean of the last hidden layer as the embedding.

Why not Wav2Vec2?  
• It led to overfitting in our experiments.  
• HuBERT gave better generalization and cleaner residuals.

# 3. Embedding Loading Logic

We created a function to load saved `.npy` embedding files and match them with their corresponding labels from `train.csv`. This ensured all embeddings were aligned with their grammar score ground truths. Missing files are skipped with a warning.

# 4. Model Training: Support Vector Regression (SVR)

We use SVR (Support Vector Regression) with an RBF kernel wrapped inside a pipeline with `StandardScaler()`.

Why SVR?  
• SVR was found to be the most stable among non-neural regressors.  
• It gives consistent and interpretable results.  
• It works well with high-dimensional feature vectors like HuBERT embeddings.

SVR Hyperparameters:  
• Kernel: 'rbf' for non-linear boundaries  
• C=10: Regularization strength  
• gamma='scale': Automatic kernel width scaling

# 5. Evaluation Metrics

On validation set, we compute:  
• Pearson Correlation – Measures linear correlation (used in competition metric)  
• MAE – Mean Absolute Error  
• RMSE – Root Mean Squared Error  
• R² Score – Explained variance

These give a full picture of how well the model fits the data.

# 6. Visualizations

We used three key plots:  
1. Residual Distribution: Shows how prediction errors are spread (ideal if centered around 0)  
2. Sorted Absolute Errors: Visualizes where model performs well and poorly  
3. Barplot of All Metrics: Quick glance at model quality from all angles

# 7. Final Prediction and Submission

The final trained SVR model is used to predict scores on the test set. Predictions are inserted into `test.csv`, and the file is saved as a Kaggle submission-ready CSV.

# 8. Experiments and Why They Were Dropped

• PCA → Performance dropped sharply due to loss of information.  
• Ridge Regression → High train score, poor test performance (overfit).  
• LightGBM → Slight overfitting; not consistent across folds.  
• SVR on Wav2Vec2 → Overfit; HuBERT was more robust.  
• Gradient Boosting → Slower and less stable.

Hence, the final solution focuses on HuBERT + SVR.

# 9. Conclusion

The pipeline is clean, interpretable, and competitive on the leaderboard. With proper visualizations, consistent metric tracking, and well-documented design, this setup is robust for further fine-tuning or ensemble experimentation.