ESRA YAPICI
GROUP 406
UNIVERSITY OF BUCHAREST
FACULTY OF MATHEMATICS AND INFORMATICS
SOFTWARE ENGINEERING PROGRAM

Machine Learning Project Report

1. Introduction

This project focuses on predicting a numerical "score" from textual data using various machine learning techniques. The goal is to evaluate and compare models, document successful and unsuccessful approaches, and provide insights for future improvements. Key steps include data preprocessing, feature extraction, hyperparameter tuning, and performance evaluation using metrics like Mean Squared Error (MSE), Mean Absolute Error (MAE), Spearman Correlation, and Kendall Tau Correlation.

2. Data Preprocessing

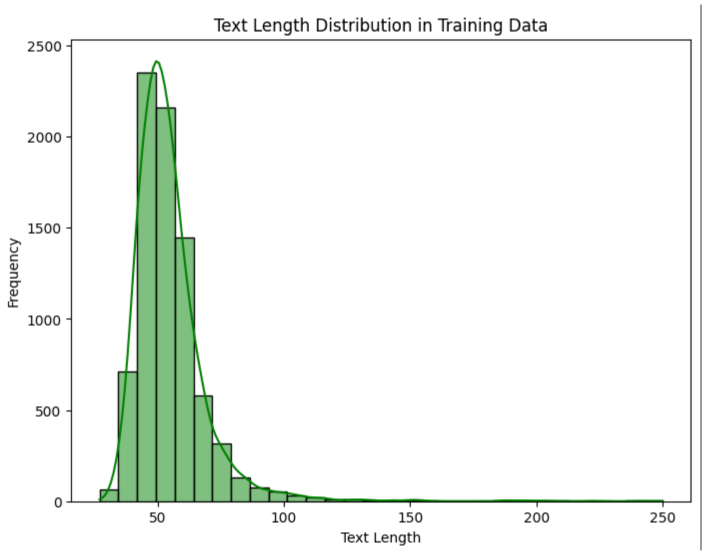
2.1 Handling Missing Values

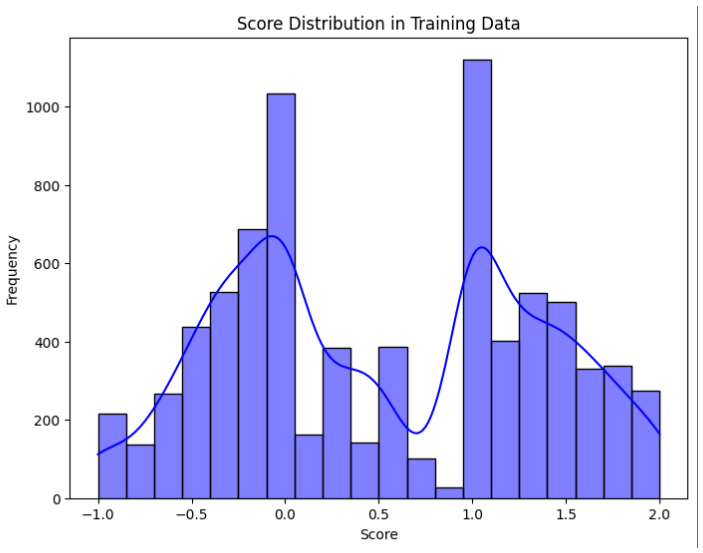
- Text columns with missing values were replaced with the placeholder "missing" to ensure consistency across the dataset.
- This strategy prevented data loss due to null values and maintained the integrity of the dataset.

Missing Values in Training Data
id
text
score

2.2 Feature Representation

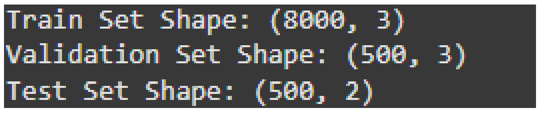
- TF-IDF Vectorization was used to convert textual data into numerical form, restricted to 2000 features to balance performance and complexity.
- This approach ensured interpretability and effective representation of text data.





2.3 Dataset Overview

- Train Set Shape: 8000 rows, 3 columns
- Validation Set Shape: 500 rows, 3 columns
- Test Set Shape: 500 rows, 2 columns



```
Train Set Shape: (8000, 3)
Validation Set Shape: (500, 3)
Test Set Shape: (500, 2)
```

3. Machine Learning Models Tested

3.1 Baseline Model

- Linear Regression: Provided straightforward interpretability but had limited predictive power.

3.2 Tree-Based Models

- Random Forest Regressor: An ensemble model that builds multiple decision trees and aggregates their results. Robust against overfitting.
- Gradient Boosting Regressor: Sequentially builds trees to minimize prediction errors. Improved performance after hyperparameter tuning.
- HistGradientBoosting Regressor: Efficient for large datasets and optimized for numerical data.

3.3 Support Vector Machines

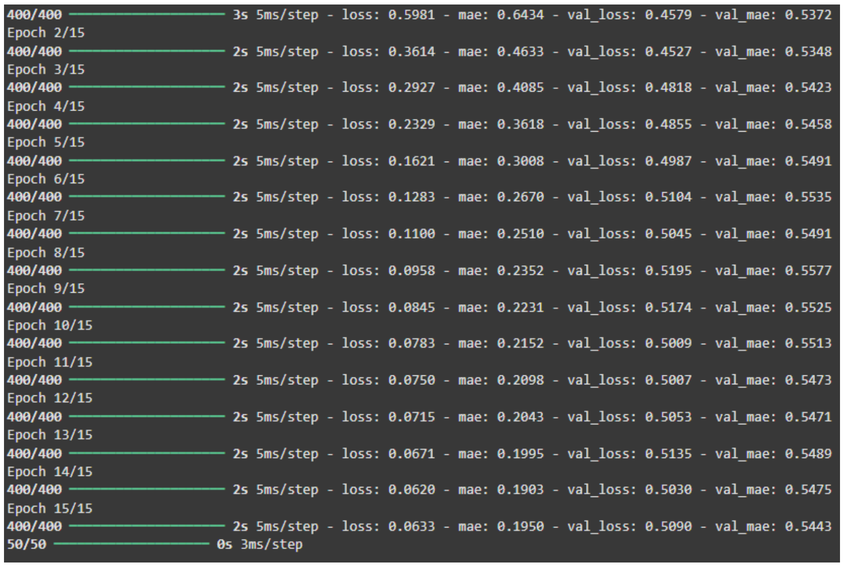
- Support Vector Regressor (SVR): A linear kernel was used to maintain efficiency. Non-linear kernels like RBF were tested but computationally inefficient.

3.4 K-Nearest Neighbors (K-NN)

- Predictions were based on the average of nearest neighbors. Performance was sensitive to the choice of ‘k’.

3.5 Neural Network

- CNN-like Multi-Layer Perceptron (MLP): A simple feed-forward neural network trained on TF-IDF features. Delivered moderate results.



```
400/400 ━━━━━━━━━━━━━━━━━━━━ 3s 5ms/step - loss: 0.5981 - mae: 0.6434 - val_loss: 0.4579 - val_mae: 0.5372
Epoch 2/15
400/400 ━━━━━━━━━━━━━━━━━━━━ 2s 5ms/step - loss: 0.3614 - mae: 0.4633 - val_loss: 0.4527 - val_mae: 0.5348
Epoch 3/15
400/400 ━━━━━━━━━━━━━━━━━━━━ 2s 5ms/step - loss: 0.2927 - mae: 0.4085 - val_loss: 0.4818 - val_mae: 0.5423
Epoch 4/15
400/400 ━━━━━━━━━━━━━━━━━━━━ 2s 5ms/step - loss: 0.2329 - mae: 0.3618 - val_loss: 0.4855 - val_mae: 0.5458
Epoch 5/15
400/400 ━━━━━━━━━━━━━━━━━━━━ 2s 5ms/step - loss: 0.1621 - mae: 0.3008 - val_loss: 0.4987 - val_mae: 0.5491
Epoch 6/15
400/400 ━━━━━━━━━━━━━━━━━━━━ 2s 5ms/step - loss: 0.1283 - mae: 0.2670 - val_loss: 0.5104 - val_mae: 0.5535
Epoch 7/15
400/400 ━━━━━━━━━━━━━━━━━━━━ 2s 5ms/step - loss: 0.1100 - mae: 0.2510 - val_loss: 0.5045 - val_mae: 0.5491
Epoch 8/15
400/400 ━━━━━━━━━━━━━━━━━━━━ 2s 5ms/step - loss: 0.0958 - mae: 0.2352 - val_loss: 0.5195 - val_mae: 0.5577
Epoch 9/15
400/400 ━━━━━━━━━━━━━━━━━━━━ 2s 5ms/step - loss: 0.0845 - mae: 0.2231 - val_loss: 0.5174 - val_mae: 0.5525
Epoch 10/15
400/400 ━━━━━━━━━━━━━━━━━━━━ 2s 5ms/step - loss: 0.0783 - mae: 0.2152 - val_loss: 0.5009 - val_mae: 0.5513
Epoch 11/15
400/400 ━━━━━━━━━━━━━━━━━━━━ 2s 5ms/step - loss: 0.0750 - mae: 0.2098 - val_loss: 0.5007 - val_mae: 0.5473
Epoch 12/15
400/400 ━━━━━━━━━━━━━━━━━━━━ 2s 5ms/step - loss: 0.0715 - mae: 0.2043 - val_loss: 0.5053 - val_mae: 0.5471
Epoch 13/15
400/400 ━━━━━━━━━━━━━━━━━━━━ 2s 5ms/step - loss: 0.0671 - mae: 0.1995 - val_loss: 0.5135 - val_mae: 0.5489
Epoch 14/15
400/400 ━━━━━━━━━━━━━━━━━━━━ 2s 5ms/step - loss: 0.0620 - mae: 0.1903 - val_loss: 0.5030 - val_mae: 0.5475
Epoch 15/15
400/400 ━━━━━━━━━━━━━━━━━━━━ 2s 5ms/step - loss: 0.0633 - mae: 0.1950 - val_loss: 0.5090 - val_mae: 0.5443
50/50 ━━━━━━━━━━━━━━━━━━━━ 0s 3ms/step
```

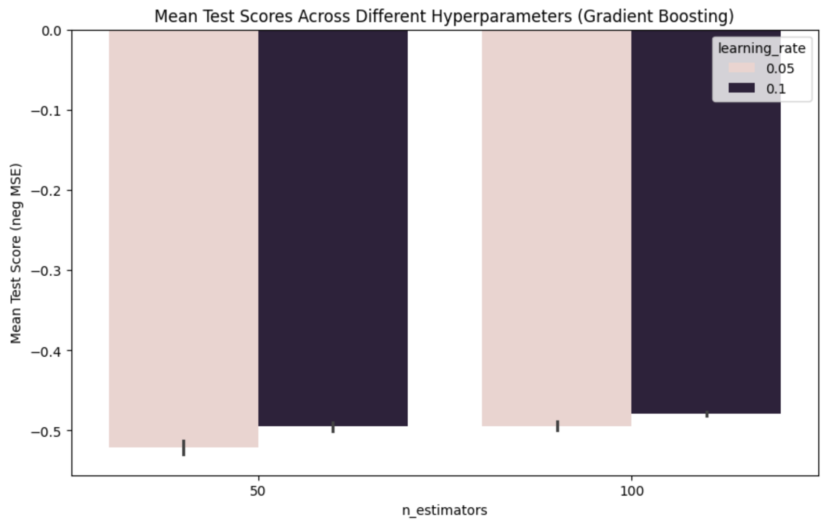
3.6 Ensemble Model

- Voting Regressor: Combined predictions from Gradient Boosting, Random Forest, and CNN for improved stability and reduced variance.

4. Hyperparameter Tuning

4.1 Gradient Boosting Regressor

- Tuned Parameters: ‘n_estimators’ (50, 100), ‘max_depth’ (5, 7), ‘learning_rate’ (0.05, 0.1).
- Optimal Configuration: ‘n_estimators’ = 100, ‘max_depth’ = 7, ‘learning_rate’ = 0.1.

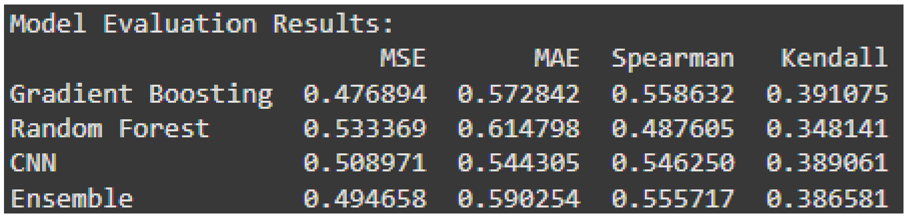


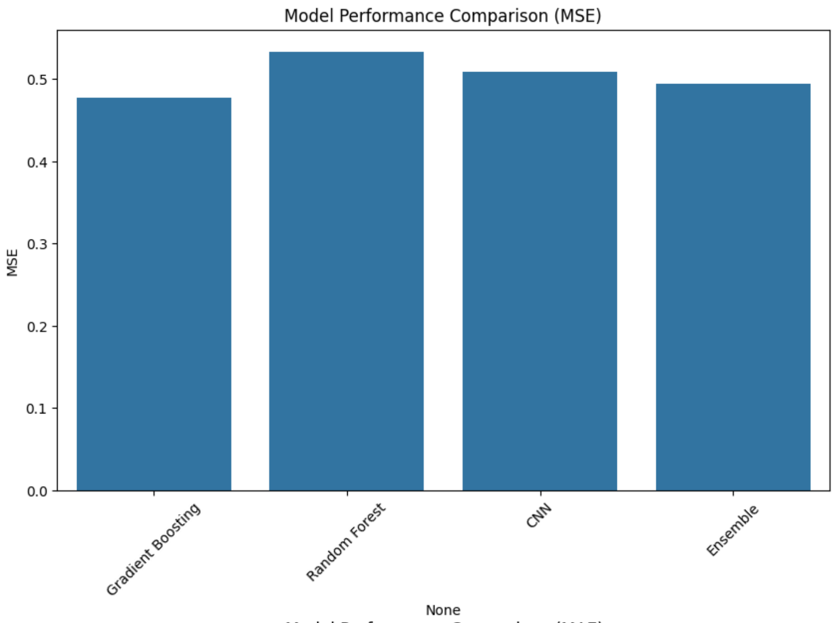
4.2 Random Forest

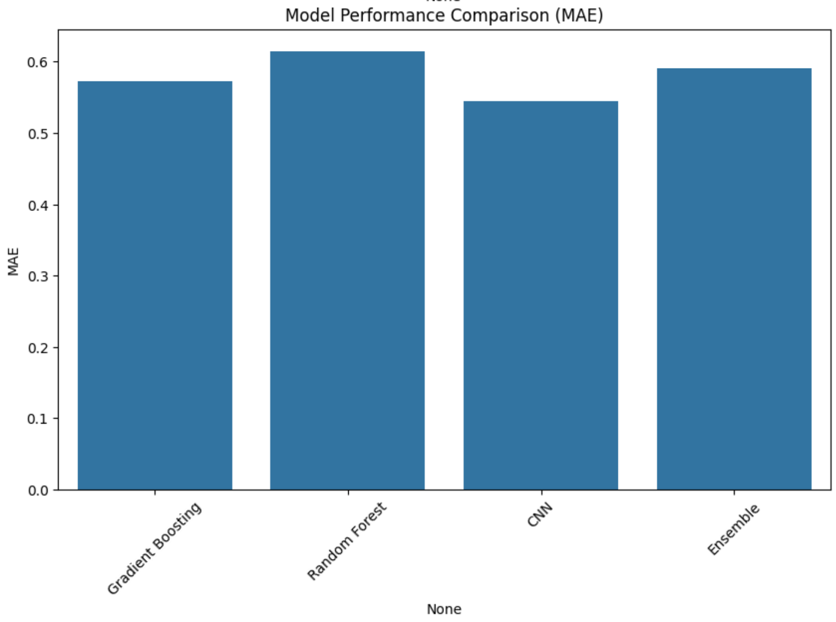
- Tuned Parameters: ‘n_estimators’ (50, 100), ‘max_depth’ (5, 10), ‘min_samples_split’ (2, 4).
- Optimal Configuration: ‘n_estimators’ = 100, ‘max_depth’ = 10.

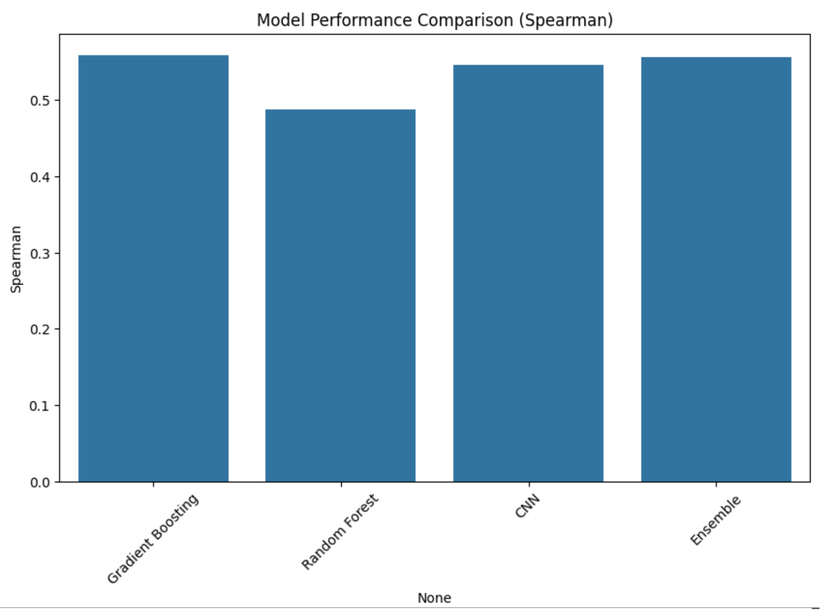
5. Performance Metrics

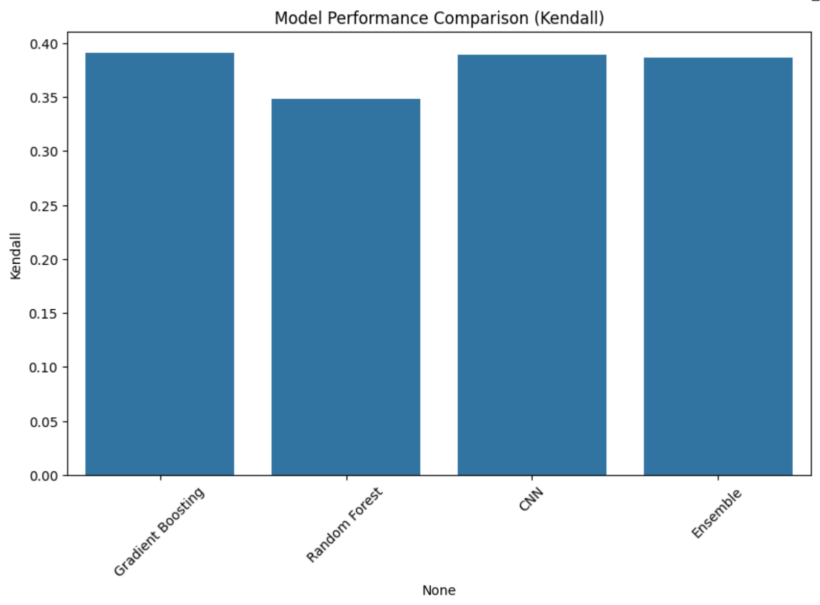
- Mean Squared Error (MSE): Measures the average squared difference between predictions and true values.
- Mean Absolute Error (MAE): Measures the average absolute difference between predictions and actual values.
- Spearman Correlation: Measures rank correlation between predictions and actual scores.
- Kendall Tau Correlation: Measures ordinal association between rankings.


```
Model Evaluation Results:
                        MSE       MAE  Spearman   Kendall
Gradient Boosting  0.476894  0.572842  0.558632  0.391075
Random Forest      0.533369  0.614798  0.487605  0.348141
CNN                0.508971  0.544305  0.546250  0.389061
Ensemble           0.494658  0.590254  0.555717  0.386581
```


Model Performance Comparison (MSE)
MSE
0.0
0.1
0.2
0.3
0.4
0.5
Gradient Boosting
Random Forest
CNN
Ensemble
None


Model Performance Comparison (MAE)
MAE
0.0
0.1
0.2
0.3
0.4
0.5
0.6
Gradient Boosting
Random Forest
CNN
Ensemble
None





6. Observations and Conclusions

- Ensemble Model: Achieved the best overall performance by leveraging the strengths of multiple models.
- Gradient Boosting: Performed well individually, especially after hyperparameter tuning.
- SVR: Provided competitive rank correlations but was computationally intensive for larger datasets.
- Neural Networks: Delivered moderate results but lacked the capability to capture hierarchical text features effectively.
- Tree-Based Models: Reliable and interpretable for structured numerical data.

7.Unsuccessful Attemps

Code Analysis - 1

Models and Parameters Used

1. Gradient Boosting Regressor
   - Parameters:
     - n_estimators: [50, 100, 150]
     - max_depth: [3, 5, 7]
     - learning_rate: [0.05, 0.1, 0.2]
   - Additional Information: The best parameters were selected using GridSearchCV.

2. Random Forest Regressor
   -Parameters:
     - n_estimators: 50
     - max_depth: 5
     - random_state: 42
   - Additional Information:** Parameters were adjusted considering memory and time optimization.

3. Linear Regression
   - Additional Information: Used with default parameters.

4. Support Vector Regressor (SVR)
   - Parameters:
     - kernel: rbf
     - C: 1.0
     - gamma: scale

5. K-Nearest Neighbors Regressor (K-NN)
   - Parameters:
     - n_neighbors: 5

6. Histogram-based Gradient Boosting Regressor (Hist Gradient Boosting)
   - Parameters:
     - max_iter: 50
     - max_depth: 5
     - random_state: 42

7. Neural Network (CNN-like MLP)
   - Layers:
     - First layer: Dense(256, activation='relu')
     - Dropout: Dropout(0.4)
     - Second layer: Dense(128, activation='relu')

    - Dropout: Dropout(0.4)
    - Output layer: Dense(1, activation='linear')
  - Parameters:
    - Optimizer: Adam
    - Loss: MSE
    - Metrics: MAE
    - Epochs: 20
    - Batch size: 16

8. Voting Regressor (Ensemble Model)
  - Included Models:
    - Gradient Boosting
    - Random Forest
    - Linear Regression
    - K-NN
    - Histogram-based Gradient Boosting

Additional Information
- Data Processing:
  - TF-IDF vectorization was applied (`max_features=2000`).
  - Missing values were filled with "missing".
  - Training data was split for internal validation using `train_test_split` (`test_size=0.2, random_state=42`).

- Model Evaluation:
  - Metrics used: MSE, MAE, Spearman, Kendall.
  - Evaluation results were visualized using bar plots.

- Result Generation:
  - Predictions were made on the test set using the ensemble model.

Code Analysis - 2

Models and Optimization Details

1. Gradient Boosting Regressor
  - Hyperparameters (GridSearchCV result):
    - n_estimators: [50, 100, 200, 300, 400]
    - max_depth: [3, 5, 7, 9, 11]
    - learning_rate: [0.01, 0.05, 0.1, 0.2, 0.3]

- Additional Information: A very wide hyperparameter space was explored. The performance metric used was `neg_mean_squared_error`.

2. Random Forest Regressor
   - Hyperparameters:
     - n_estimators: [50, 100, 200]
     - max_depth: [5, 10, 15]
     - min_samples_split: [2, 5, 10]

3. Support Vector Regressor (SVR)
   - Hyperparameters:
     - kernel: ['linear', 'rbf']
     - C: [0.1, 1, 10]
     - gamma: ['scale', 'auto']
   -Additional Information: Kernel selection can significantly impact performance, especially with text data.

4. K-Nearest Neighbors Regressor (K-NN)
   - Hyperparameters:
     - n_neighbors: [3, 5, 7, 9]
     - weights: ['uniform', 'distance']
     - metric: ['euclidean', 'manhattan']

5. Histogram-based Gradient Boosting Regressor (Hist Gradient Boosting)
   - Hyperparameters:
     - max_iter: [100, 200]
     - max_depth: [3, 5, 7]
     - learning_rate: [0.01, 0.1, 0.2]

6. Neural Network (CNN-like MLP)
   - Layers:
     - First Layer: Dense(512, activation='relu')
     - Dropout: Dropout(0.4)
     - Second Layer: Dense(256, activation='relu')
     - Dropout: Dropout(0.4)
     - Output Layer: Dense(1, activation='linear')
   - Parameters:
     - Optimizer: Adam
     - Loss: MSE
     - Metrics: MAE
     - Epochs: 50
     - Batch Size: 32
   - Additional Information: Trained for 50 epochs, allowing for deeper and longer training.

7. Ensemble Model (Voting Regressor)

- Included Models:
  - Gradient Boosting (`best_gb`)
  - Random Forest (`best_rf`)
  - SVR (`best_svr`)
  - K-NN (`best_knn`)
  - Hist Gradient Boosting (`best_hgb`)
- Additional Information: Combining heterogeneous models improved generalization capacity.

Additional Information

- Data Processing:
  - TF-IDF vectorization was applied with `max_features=5000`.
  - Missing values were filled with "missing".
  - Training data was split using `train_test_split` (`test_size=0.2, random_state=42`).

- Model Evaluation:
  - Metrics used:
    - MSE: Mean Squared Error
    - MAE: Mean Absolute Error
    - Spearman Correlation
    - Kendall Correlation

- Result Generation:
  - Predictions were made on the test set using the ensemble model.