Journal of Research High School

Vol. 2026 (7) 552 – 560
ISSN 2688-3651 [online]

A Comparative Study of Regression and Classification Models for Text-Based Scoring

Yatharth Sathya[1] *

[1]Archbishop Mitty High School, San Jose, CA, USA
*Corresponding Author: yatharth.sathya@gmail.com

Advisor: Odysseas Drosis, od84@cornell.edu

Received August 22, 2025; Revised February 19, 2026; Accepted March 26, 2026

Abstract

Automated evaluation of textual responses has remained a challenge, as traditional scoring methods rely heavily on subjective human judgment. Human scoring can be inconsistent and difficult to scale, due to factors such as inherent biases and fatigue. This study examined whether automated models can be used to predict numerical scores from short text responses, a task that was traditionally dependent on human evaluation. Using a comparative approach, several automated scoring methods were evaluated on short speech samples to assess their ability to score these text sentences. In summary, the findings suggested that machine learning has promise in the subfield of automated text assessment, but further work was needed to improve reliability and consistency in these scoring systems.

Keywords: Machine learning, Automated text scoring, Text embeddings

1. Introduction

In competitive environments where short speeches and presentations are scored: such as academic contests, evaluations, or professional events, accurate and fair scoring was essential. These settings often rely on human judges to assess performance, but human evaluation comes with unavoidable challenges: subjectivity, variability in scoring styles, and differences in experience can all impact results. Even well-trained judges may be influenced by personal preferences or fatigue, leading to inconsistent assessments and potential unfairness.

To address these challenges, researchers have explored automated text scoring (ATS) systems over the past several decades. Early approaches, such as rule-based or statistical methods, analyzed surface-level features like word count, sentence length, and spelling to generate scores (Stevenson, 2013). While these methods provided faster scoring than humans, they could not evaluate deeper elements of writing such as argument quality, style, or emotional impact. Later, neural network–based systems improved automated scoring by learning complex patterns in text, as seen in work on Automatic Text Scoring using Neural Networks (Alikaniotis et al., 2016). In recent years, advances in natural language processing (NLP) and machine learning (ML) have opened the door to automated text evaluation, including transformer-based models such as BERT, offering the possibility of faster, more consistent scoring. By training models to analyze speech transcripts and predict scores, competitions could reduce reliance on large pools of human judges, minimize bias, and provide rapid feedback to participants.

Despite these advances, most prior research has focused on long-form essays or large datasets. This represents a significant gap, especially in competitive and educational settings where short-text scoring is common and consistency is essential. The critical question is whether ML models can capture enough nuance in short text (2-3 lines in our experiment) to evaluate performance as reliably as a human.

Building on these prior works, this study investigates whether ML models can predict numerical scores for short text responses. Using a comparative approach, multiple models were trained and evaluated to analyze how model complexity affects scoring performance. The analysis focuses on predicting scores based on multiple criteria, including

Vol. 2026 (7) 552 – 560
ISSN 2688-3651 [online]

the text's purpose, content development, emotional appeal, and structural quality. By exploring these factors, this study aims to assess whether ML can provide reliable and consistent scoring, offering a practical tool to support human judges in competitive environments.

In recent years, advances in NLP and ML have opened the door to automated text evaluation, offering the possibility of faster, more consistent scoring. By training models to analyze speech transcripts and predict scores, competitions could reduce reliance on large pools of human judges, minimize bias, and provide rapid feedback to participants. However, the critical question was whether ML models can capture enough nuance in short text (2-3 lines in our experiment) to evaluate performance as reliably as a human.

One of the main references in this project was a research paper called *Advances in Debating Technologies: Building AI That Can Debate Humans*. It was written by Roy Bar-Haim, Liat Ein-Dor, Matan Orbach, Elad Venezian, and Noam Slonim as part of IBM's research program. The paper focuses on the idea of developing artificial intelligence (AI) that was able to debate against its human counterparts. As AI continues to grow and improve, especially in the area of NLP, researchers have started exploring something called *computational argumentation*. This encompasses a subfield of NLP that looks at how computers can be trained to understand, analyze, and even create arguments like a human (Bar-Haim et al., 2021). This reference paper further goes on to prove that people have already started tackling the big question of whether AI can truly engage in argument-based tasks, not just by understanding language, but also using logic and reasoning in ways comparable to humans. This makes it an important reference for our own project, which focuses on using AI to evaluate text.

Another of the references that was looked at while doing this experiment was a paper called *Automatic Text Scoring Using Neural Networks*. It was written by Dimitrios Alikaniotis, Marek Rei, and Helen Yannakoudakis as part of a study done by the University of Cambridge. This paper talks about using Neural Networks to provide scoring to certain texts: an automatic text scorer powered by neural networks (Alikaniotis et al., 2016). Therefore, the decision was made to extend this theory, from neural networks to multiple types of ML models, such as Linear Regressors, Decision Trees, and more. This makes it an important reference for this specific project, which focuses on using AI to evaluate text, as this paper elaborates on the latter's foundational work.

BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding was utilized heavily while completing this paper. It was written by Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova as part of a study completed by Google. This paper introduces a new language representation model called BERT. Unlike recent language representation models, BERT was designed to pre-train deep bidirectional representations from unlabeled text by jointly conditioning on both left and right context in all layers (Devlin et al., 2018). BERT has been extensively used in Automated Essay Scoring (AES) research, with studies showing how pre-trained models like BERT can be applied to text evaluation tasks. In addition to that, the sentence transformer model used (all-MiniLM-L6-v2) was based on BERT's architecture, showcasing how transformer-based embeddings are effective for text analysis.

Ultimately, the core hypothesis was as follows: a model can be trained from a small dataset to predict scores (graded on the text's purpose, its ability to develop content, its emotional appeal to the audience and the structure of the text) based on two to three sentences of text.

2. Materials and Methods

The experiment began with a sentence transformer model, "all-MiniLM-L6-v2", which maps sentences to a 384 dimensional dense vector space. This was utilized to ensure that our text could be easily accessible to the models.

In this experiment, eight ML models were tested: four for regression (predicting numerical values) and four for classification (predicting labels or categories). Each model has different strengths, weaknesses, and parameters that influence its behavior and performance. Both regression and classification models were applied for multiple purposes. Primarily, the regression models predicted numerical scores assigned to the text responses. This approach allows the model to capture subtle differences between responses. For example, if one sentence was predicted as 18 and another as 19, then a new sentence that falls in between can be assigned an intermediate score, such as 18.5. Classification models were also applied to explore an alternative perspective, where scores are treated as discrete categories rather

Vol. 2026 (7) 552 – 560
ISSN 2688-3651 [online]

than flexible, numerical values. This approach can sometimes improve stability and robustness, especially during instances where a human's scoring would be inconsistent or when small differences in numerical scores are less meaningful.

Comparing regression and classification approaches, especially during this experiment was important in this context, because it highlights how the choice of models affects the ability to replicate human judgment. While regression can capture subtle differences, classification may better handle subjective variability and simplify the automated task. By examining both, several metrics can be evaluated in ATS, providing insights into how ML can most effectively support, or even automate traditional human evaluation.

Our first model was the linear regressor as shown in Figure 1. This model tries to find a straight-line relationship between the input features and the output. It was a good starting point as a model, with a mean squared error (MSE) of 7.4 for testing and 0.006 for training. However, because it assumes that the relationship between the data was linear, it may not perform well if the actual pattern in the data was more complex. In our case, it performed very well on the training data but didn't do as well on the test data, which suggests it may have seen overfitting. This proves that the model will memorize everything about the training data, including noise, rather than patterns.

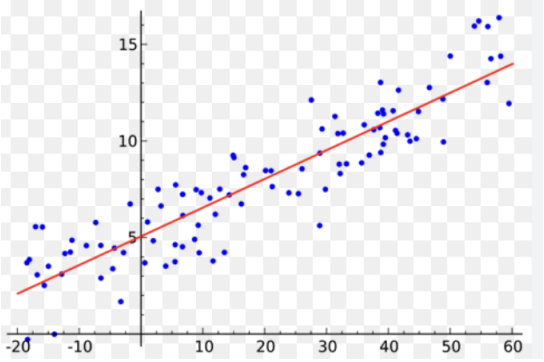

Figure 1. Linear Regression Sample. Cite: Tahir, 2018. A diagram illustrating how linear regression fits a straight line to predict numerical scores from text embeddings, the simplest regression model tested in this study.

Next as shown in Figure 2, a decision tree regressor was used, which works by splitting the data into smaller and smaller groups based on decisions, like a flowchart. This model can handle more complex patterns than linear regression. It performed better on the test data than the linear model, showing that it was better at capturing the structure of the data. However, decision trees can sometimes become too specific to the training data, especially if they grow too deep, which can hurt their ability to generalize to new data. This proves the criticality of the adjustment of the max_depth hyperparameter. Max_depth prevents the tree from growing too deep and potentially overfitting the training data (ultimately the decision was to set max_depth to 16). The dataset is small, so complex models like Decision Trees and Random Forests have relatively few examples to generalize from, making it easy for them to memorize training samples rather than learn patterns. The scoring range is narrow (15–20), meaning small differences in text quality map to very compressed score differences, which makes it harder for models to learn truly generalizable boundaries. Short text (2–3 sentences) provides limited features, so models may latch onto surface-level word patterns specific to the training set rather than deeper rhetorical qualities. This has an effect in the field on automatic text scoring in practice. A model trained on one competition's speech samples, there is a possibility that it may score very differently when applied to a new group of students or a different topic area. In high-stakes settings like academic competitions, an overfitted model could systematically disadvantage participants whose writing style differs from the training data, even if their content quality is comparable. Overfitted models are also harder to audit, since their decisions reflect memorized examples rather than interpretable scoring criteria, making it difficult for judges or participants to understand or contest a score. Deployment would likely require retraining the model on new data each competition cycle, which raises questions about consistency across years. This clearly illustrates why future models will need to take into account different writing styles to ensure fairness across dimensions,

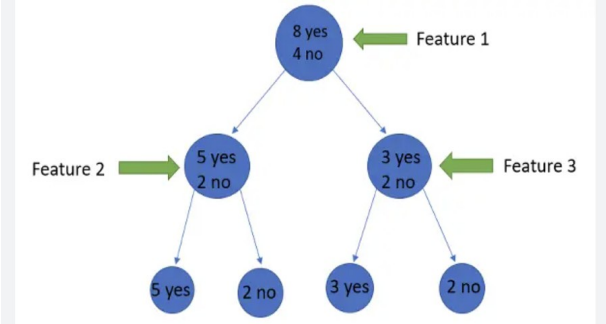

Figure 2. Decision Tree Sample. Cite: Parvekar, 2023. A diagram illustrating how a decision tree regressor splits data into progressively smaller groups based on feature values to predict scores, the approach that outperformed linear regression on test data in this study.

Vol. 2026 (7) 552 – 560
ISSN 2688-3651 [online]

through larger, more diverse datasets.

Consequently, the MLP Regressor (shown in Figure 3) was tested, following the same neural network architecture. However, this regressor was designed for predicting continuous values instead of class labels. It could model highly complex relationships in the data, and was a neural network, which worked with weights, rather than predicting decisions through a model like a tree. The MLP Regressor was the worst performing model, as it was the only model to have a MSE more than one in training (3.11), as well as a very high MSE in testing (7.8). Figure 3 represents a diagram of a neural network.

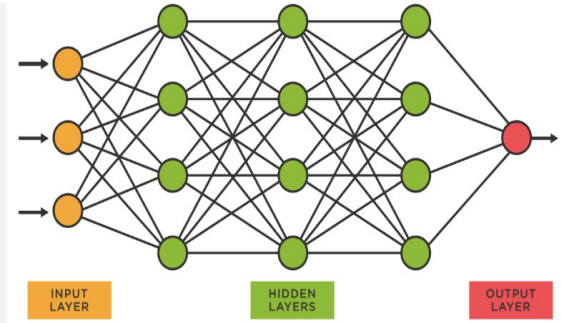

Figure 3: Neural Network Sample. Cite: Nfa.Ai, 2023. A diagram illustrating the architecture of a neural network, representing the structure underlying the MLP Regressor.

To address this, a random forest regressor (shown in Figure 4) was simultaneously tested, which combines many decision trees and averages their predictions. This helps reduce the chance of overfitting, usually providing more stable and accurate results. In our experiment, the random forest regressor performed the best among the regression models. It had a good balance between learning from the training data and still performing well on the testing data, making it a strong choice for regression tasks. An additional hyperparameter, n_estimators (or the number of decision trees in the random forest) was subsequently adjusted to ensure that our regressor was optimal. Figure 4 represents a sample diagram of a random forest classifier.

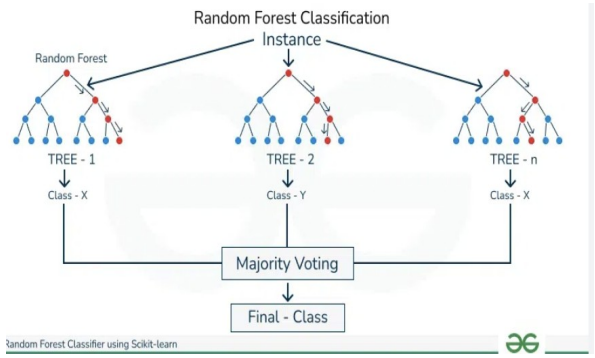

Figure 4. Random Forest Sample. Cite: GeeksforGeeks, 2025. A diagram illustrating how a random forest combines multiple decision trees and averages their predictions to improve accuracy and reduce overfitting.

After this, the focus of the experiment shifted over to classification models. The first test began with a linear classifier, which tries to separate data into categories using a straight line or boundary. Similar to linear regression, it was simple and fast, but it might miss complex patterns in the data. It performed reasonably well, but was not the most accurate model.

The decision tree classifier works similarly to the regressor version, splitting the data based on feature values to reach a decision. It can handle more complex data but, like before, risks overfitting if the tree was too deep. Its performance showed that it could learn the training data well but struggled a bit more with the test data.

Following this, the MLP Classifier was tested, which was a type of neural network that can capture complex, non-linear relationships in the data. The MLP Classifier processes inputs through multiple layers of interconnected nodes, each applying weights and activation functions to transform the data. This allows the model to detect patterns that simpler models might miss. Through the testing phase, the MLP Classifier performed similarly to the other models. The eventual conclusion was that while it could achieve high accuracy, it was also more computationally intensive than simpler models, and thus had a larger inference time.

Finally, the random forest classifier combines multiple decision trees and predicts the most common class among them. This makes it more robust and less likely to overfit. It performed very well on the training data, but this could be a sign that it memorized the training set too closely. While it still did better than the single decision tree on the test set, the large gap between training and test performance suggests some overfitting.

In ML, accuracy was a metric commonly used in classification tasks, where the model predicts discrete categories or labels. Accuracy measures the percentage of predictions that exactly match the true labels. For example, if a model correctly classifies 90 out of 100 samples, it has 90% accuracy.

However, in regression tasks, the goal was fundamentally different. Regression models predict numbers that can have decimals (e.g. 17.03, 19.32), rather than choosing from fixed categories or labels, like a classifier does (in the

Vol. 2026 (7) 552 – 560
ISSN 2688-3651 [online]

case of our experiment, this would be whole numbers). This leads to the fact that exact matches are extremely rare. The model's predictions are usually close approximations rather than exact values. For example, if the true value was 5.00 and the model predicts 5.03, it's considered a good prediction, but it's not exact, so accuracy doesn't apply. Also, accuracy doesn't make sense in this context. If accuracy was used as the percent of exactly correct predictions, it would almost always be zero, even for well-performing models.

Instead, regression models are evaluated using error-based metrics that measure how close the predictions are to the actual values. The Mean Squared Error (MSE) is the average of squared differences between predicted and actual values. Figure 5 represents the MSE equation measuring the average squared difference between predicted and actual values, which is a standard equation when evaluating ML models. In Figure 5 MSE metric, y represents the true score, ŷ represents the predicted score, and n represents the total number of samples. Squaring the differences ensures that larger errors are penalized more heavily than smaller ones. A lower MSE therefore indicates that the model's predictions are closer to human judgment. MSE is particularly appropriate for this study because regression models often predict decimal values (e.g., 18.3 or 19.1), making exact matches unlikely and accuracy metrics unsuitable.

$$\text{MSE} = \frac{1}{n}\sum_{i=1}^{n}(y_i - \hat{y}_i)^2$$

Figure 5: Mean Squared Error Equation. Cite: OpenAI. (2025). GPT-5. labs.openai.com. The Mean Squared Error (MSE) equation used to evaluate all regression models in this study, where ŷ represents the predicted score, y represents the true score, and n represents the total number of samples.)

Additionally, when analyzing the classification models, the addition of classification reports as an evaluation metric was implemented. In the classification results, four metrics were utilized: precision, recall, f1 score and support. Equation 1 shows the equation for the precision metric, which measures how often the model's predicted scores were correct. It is essentially the number of true positives divided by the total number of predictions made for a given score. High precision indicates that when the model assigns a score, it is usually correct. This is important in ATS, to ensure that scores are not assigned too easily.

$$E(Precision) = E(True\ Positive) / (E(True\ Positives) + E(False\ Positives)) \quad (1)$$

Equation 1: Precision Equation. The precision equation used to evaluate classification models, measuring how often a model's predicted score category was correct, calculated as true positives divided by total positive predictions.

Equation 2 illustrates Recall, which is the amount of true positive values that the model actually identified compared to all of the values that the model detected as positive, reflecting its ability to capture relevant cases. It is calculated as the number of true positives divided by the total number of actual instances of that score. High recall indicates that the model successfully captures most responses that truly belong to a particular score category, reducing the risk of under-scoring strong responses.

$$Recall = TP / (TP + FN) \quad (2)$$

Equation 2. Recall Equation. Variables: TP - True Positives, FN - False Negatives. The recall equation used to evaluate classification models, measuring how many truly correct score instances the model successfully identified, calculated as true positives divided by true positives plus false negatives.

Equation 3 shows the F1 score, which combines precision and recall into a single metric by computing their harmonic mean. This metric was especially useful because it balances both correctness and completeness. A model with a high F1 score demonstrates that it not only assigns scores accurately but also consistently identifies all relevant cases. This balance was critical in scoring tasks where both fairness and reliability matter. The F1 score was necessary, as it prevents a model from looking good just because a singular metric was displayed as high. In other experiments, such as datasets where one class dominates, accuracy can be misleading. F1 focuses only on the positive class and how well it was assessed by the model. Since both the precision and recall variables are critical to this experiment, as it was necessary that the model simply just does not give out higher scores, the F1 score was used to detail this.

$$F1 = 2 * ((Precision * Recall)/(Precision + Recall)) \quad (3)$$

Equation 3. F1 Score Equation. Cite: OpenAI. (2025). GPT-5. labs.openai.com. The F1-score equation used to evaluate classification models, representing the harmonic mean of precision and recall to balance both correctness and completeness in a single metric.

Vol. 2026 (7) 552 – 560
ISSN 2688-3651 [online]

The Support metric represents the number of actual samples for each score category in the dataset. While support was not a performance metric, it provides important context for interpreting results. A high F1 score on a category with very few samples may be misleading, whereas consistent performance on categories with high support suggests better generalization. For instance, a good performance on 5 examples doesn't mean the model will generalize well. In this experiment, the support variable has been used to provide context, showcasing how the model categorized different scores based on the text responses (for instance, the MLP Classifier assigned 28 instances of these text samples a score of 18.0)

In addition, macro average treats all classes equally (unweighted mean of per-class metrics), ideal for balanced importance, while weighted average considers each class's frequency (support) in the dataset, giving more influence to larger classes, making it better for imbalanced data to reflect overall model performance.These metrics reflect the continuous nature of the prediction task and offer a much more meaningful assessment of model performance than accuracy ever could in regression.

The dataset used in this study consists of short, 2–3 line speech samples, each annotated with a numerical score ranging from 15 to 20. These texts resemble concise spoken-word pieces and address a wide range of themes, including leadership, education, mental health, social awareness, and self-expression. The scoring criteria consists of several qualitative dimensions: the text's purpose, development of content, emotional appeal, and structural coherence. Each text was designed to convey a message quickly and effectively, making the dataset both semantically rich and compact. The dataset was created to simulate competitive or performance-based settings, where quick, expressive writing was judged on both rhetorical impact and content development. The dataset was divided into an 80/20 train test split, because the models were required to be able to work on unseen examples, while also ensuring necessary training data for the models. The dataset's characteristics make it suitable for training ML models aimed at replicating or approximating human evaluative judgment on short-form, high-stakes written content.

One example from the dataset was: If education was meant to help us grow, why do so many of us feel like we're drowning? It's time to value learning over letter grades—and students over statistics. This sentence received an 18, because the emotional appeal fell a bit short (due to the lack of a personal anecdote/incident) and the content was not fully developed.

Language models generate outputs based on probabilities, and how these outputs are chosen depends on whether the task requires determinism or creativity. In deterministic settings, like solving equations or calculating financial interest rates, the model should produce the same output for a given input every time. These tasks have only one correct answer, so it becomes crucial to eliminate randomness. To achieve consistent behavior, the model was configured with temperature set to zero, eliminating randomness and forcing it to choose the most probable option at each step (Troshin et al., 2025).

In contrast, non-deterministic outputs are necessary for more open-ended tasks like writing stories, composing music, or holding a natural conversation. These activities have many possible correct answers. By increasing the temperature variable, controlled randomness was introduced to the environment, allowing the model to explore different word choices and produce varied outputs, even from the same prompt. Everything depends on the temperature, because higher values make the output more creative, while lower values keep it more predictable.

A common method for generating text was the greedy procedure, where the model picks the highest-probability token at each step without considering long-term consequences. This method was fast and simple, but may lead to repetitive or bland text because it lacks exploration. To improve quality while still managing computation, techniques like top-k sampling can be used (Chen et al., 2024). This limits the model to choosing from the top "k" most likely words, allowing some variability while maintaining reasonable coherence. A larger top-k allows more exploration but increases inference time.

Another factor in output quality was how sequences, not just individual words, are evaluated. A word with a high initial probability may lead to a weak sentence, while a less likely word might start a stronger overall sequence. Therefore, models often apply sequence-level scoring, adjusting for factors like length penalties to avoid outputting extreme outputs and ensuring efficiency.

Finally, to make models more efficient for deployment, techniques like quantization are applied. Quantization reduces the precision of the model's internal weights, rounding them to fewer decimal places, to speed up computation

Vol. 2026 (7) 552 – 560
ISSN 2688-3651 [online]

and reduce memory usage (Jin et al., 2023). This results in faster inference times, with a small trade-off in accuracy or output quality. This leads to higher customer satisfaction at some points, but it must be managed well.

3. Results

MSE was utilized as a metric for regression models because these regression models predict values in between the predefined scores (e.g. decimal values such as 15.33), and were much less likely to have a pure accuracy score. Meanwhile, classifiers always attempt to classify the texts into categories, which were given as the predefined scores (e.g. 16, 18). Classification reports are a means to elaborate on these results.

Table 1 summarizes the training and testing Mean Squared Error (MSE) for all regression models evaluated in this study.

Table 1 Regression Model Performance: Training vs. Testing MSE:

Model	MSE (Test)	MSE (Train)
Linear Regression	7.4	0.006
Decision Tree Regressor	2.05	0.06
MLP Regressor	7.8	3.77
Random Forest Regressor	1.07	0.11

MSE references Mean Squared Error. See Figure 5 for the equation

Table 2. Random Forest Classifier Performance on Test Data.

Random Forest Classifier	precision	recall	F1-Score	Support
15.0	0.00	0.00	0.00	2
17.0	1.00	0.29	0.44	7
18.0	0.50	0.46	0.48	28
19.0	0.44	0.72	0.55	39
20.0	0.27	0.12	0.16	26

Table 3. MLP Classifier Performance on Test Data.

MLP Classifier	precision	recall	F1-Score	Support
15.0	0.00	0.00	0.00	2
17.0	0.40	0.29	0.33	7
18.0	0.38	0.50	0.43	28
19.0	0.50	0.41	0.45	39
20.0	0.32	0.35	0.33	26

Table 4. Decision tree Classifier Performance on Test Data.

Decision Tree Classifier	precision	recall	F1-Score	Support
15.0	0.00	0.00	0.00	0
16.0	0.00	0.00	0.00	2
17.0	0.40	0.29	0.33	10
18.0	0.38	0.50	0.43	31
19.0	0.50	0.41	0.45	31
20.0	0.32	0.35	0.33	28

Table 5. Linear Classifier Performance on Test Data.

Decision Tree Classifier	precision	recall	F1-Score	Support
15.0	0.00	0.00	0.00	2
17.0	0.40	0.29	0.33	7
18.0	0.36	0.36	0.36	28
19.0	0.50	0.47	0.47	39
20.0	0.32	0.42	0.37	26

Classification Model Performance on Test Data (Precision, Recall, F1-Score, and Support):

As shown in Table 2, the Random Forest Classifier achieved the strongest overall classification performance on the test dataset, particularly in mid-range score categories.

The results from the Random Forest Classifier showcased a relatively strong f1-score range (between 0.4 to 0.6 on the median values). This demonstrated the ability for the model to characterize the results relatively well.

The MLP Classifier had a much lower skew, with the results decreasing significantly from the previous model as seen in Table 3.

The Decision tree Classifier shown in Table 4 had a relatively good range of f1 scores, yet the scores were still well under that of the Random Forest Classifier.

The Linear Classifier shown in Table 5 also had good f1 scores, but they were still significantly under the 0.5 baseline mark. Table 6 compares training and testing accuracy across all classification models, highlighting varying degrees of overfitting.

Overall, this experiment showed that more complex models like random forests tend to perform better when tuned properly, especially when the data relationships aren't simple. However, simpler models like linear regression and classifiers can still be useful, especially when the data was easy to separate or when fast, interpretable results are required. Understanding model behavior, like how they react to different data or how deeply they're allowed to learn (as in max depth), was key to choosing the right one for a given task.

The Linear Regression showed the highest error on both training and testing sets. The Decision Tree Regressor had very minimal error on training data, indicating overfitting, and performed moderately on the

Table 6. Classification Model Performance: Training vs. Testing Accuracy.

Model	Accuracy (Test)	Accuracy (Train)
Linear Classifier	0.36	0.67
MLP Classifier	0.38	0.98
Decision Tree Classifier	0.36	0.99
Random Forest Classifier	0.45	0.99

Vol. 2026 (7) 552 – 560
ISSN 2688-3651 [online]

test set. Although some of the models overfit, like the decision tree, there is no particular reason for why this model specifically overfits. However, after experimentation, the primary hypothesis is that the max_depth hyperparameter is the primary cause of this overfitting. When max_depth was adjusted in future experiments, the training error went up, while the testing error went down, proving overfitting. This affects model reliability because it makes the model retain specifics of the dataset, rather than focusing on the patterns. That decreases its generalizability in practical applications, because it will not be able to process other datasets well.

The Random Forest Classifier also had zero training error and a slightly higher test error than the decision tree, confirming some overfitting but competitive performance. The Random Forest Classifier was the best performing model. The Random Forest Classifier was a ML model that consists of a set of decision trees. Since the Random Forest Classifier has multiple decision trees, it generally gives lower MSE values. Both the Random Forest models, as well as the decision trees had their hyperparameters specifically tuned, especially with the Random Forest models. In the Random Forest, the max_depth and n_estimators were fine-tuned until optimal results were received (with an optimal max_depth set to 16 and an n_estimators value set to 50). This showcased that the differences in values can furthermore lead to different MSE/Accuracy rates, by allowing the model more trees to predict answers (n_estimators), as well as increasing the depth of those trees (max_depth). This can solve the overfitting problem, because the model will learn to memorize patterns in the data, rather than not memorizing enough, or memorizing noise.

However, in regression tasks, the goal was fundamentally different. Regression models predicted numbers that can have decimals (e.g. 17.03, 19.32), rather than choosing from fixed categories or labels, like a classifier does (in the case of our experiment, this would be whole numbers). This led to the fact that exact matches are extremely rare. The model's predictions were usually close approximations rather than exact values. For example, if the true value was 5.00 and the model predicts 5.03, it's considered a good prediction, but it's not exact, so accuracy doesn't apply. Also, accuracy doesn't make sense in this context. If accuracy were to be used as the percent of exactly correct predictions, it would almost always be zero, even for well-performing models.

Alternatively, the regression models were evaluated using error-based metrics that measured how close the predictions were to the actual values. The MSE was the average of squared differences between predicted and actual values.

These metrics reflected the continuous nature of the prediction task and offered a much more meaningful assessment of model performance than accuracy ever could in regression.

4. Conclusion

This paper examined whether ML models could have predicted numerical scores from short text sequences to automate evaluation in competitive settings. Eight models, four regression and four classification, were trained on text embeddings from scored speech samples. Tree-based models performed best, with the Random Forest Regressor achieving an MSE of 1.07, though it showed signs of overfitting. Linear models generalized more consistently.

Overall, the results suggest ML can support preliminary, more objective text scoring, but careful tuning and validation are needed before use in high-stakes contexts.

1. Can we apply this technology to other fields such as art or music?
2. Can we apply this technology to other languages (e.g. Mandarin, Hindi)?
3. Can this technology capture metaphorical expressions?

References

Alikaniotis, D., Yannakoudakis, H., & Rei, M. (2016). Automatic Text Scoring Using Neural Networks. Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers), 715–725. https://doi.org/10.18653/v1/P16-1068

Bar-Haim, R., et al. (2021). Advances in Debating Technologies: Building AI That Can Debate Humans. 1–5. https://doi.org/10.18653/v1/2021.acl-tutorials.1

Vol. 2026 (7) 552 – 560
ISSN 2688-3651 [online]

Chen, S., Hagrass, O., & Klusowski, J. M. (2024, October 4). *Decoding Game: On Minimax Optimality of Heuristic Text Generation Strategies*. ArXiv.org. https://arxiv.org/abs/2410.03968

Devlin, J., et al. (2019). BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding. ArXiv:1810.04805 [Cs]. https://arxiv.org/abs/1810.04805

Do, H., Ryu, S., & Lee, G. G. (2025). *Teach-to-Reason with Scoring: Self-Explainable Rationale-Driven Multi-Trait Essay Scoring*. ArXiv.org. https://arxiv.org/abs/2502.20748

GeeksforGeeks. (2025, September 13). *Random Forest Classifier using Scikitlearn*. https://www.geeksforgeeks.org/dsa/random-forest-classifier-using-scikit-learn/

Jin, J., et al. (2023, June 4). *OWQ: Outlier-Aware Weight Quantization for Efficient Fine-Tuning and Inference of Large Language Models*. ArXiv.org. https://arxiv.org/abs/2306.02272

Nfa.Ai. (2023, January 12). *How a neural network makes decisions*. Medium. https://medium.com/@nfalabs/how-a-neural-network-makes-decisions-79ddd9573f7

OpenAI. (2025). GPT-5. labs.openai.com

Ormerod, C. M., Malhotra, A., & Jafari, A. (2021). Automated essay scoring using efficient transformer-based language models. ArXiv.org. https://arxiv.org/abs/2102.13136

Parvekar, V. (2023, November 27). *Decision Tree in Machine Learning.* Medium. https://medium.com/@parvekarvedant24/decision-tree-in-machine-learning-efd47509813

Stevenson, M. (2013). Shermis, M.D. & Burstein, J. (Eds) (2013). Handbook of Automated Essay Evaluation: Current applications and new directions. Routledge: New York and London. ISBN-10: 0415810965. *Journal of Writing Research*, *5*(2), 239–243. https://doi.org/10.17239/jowr-2013.05.02.4

Tahir, B. (2018, November 24). *A Quick Guide To Using Regression With Image Data In Fastai*. Medium. https://btahir.medium.com/a-quick-guide-to-using-regression-with-image-data-in-fastai-1 17304c0af90

Troshin, S., et al. (2025). *Control the Temperature: Selective Sampling for Diverse and High-Quality LLM Outputs*. ArXiv.org. https://arxiv.org/abs/2510.01218