Predicting Student Exam Scores Using Tabular Machine Learning

Lars Moen Storvik
FEUP
University of Porto
Porto, Portugal
up202508437@up.pt

Tina Kovačević
FEUP
University of Porto
Porto, Portugal
up202501724@up.pt

Zakariea Sharfeddine
FEUP
University of Porto
Porto, Portugal
up202501730@up.pt

***Abstract*—This paper addresses the Kaggle Playground Series S6E1 competition: predicting student exam scores from demographic, academic, and behavioral features. We develop a tabular learning pipeline with feature engineering, cross-validation, and Bayesian hyperparameter optimization, and we benchmark both traditional regressors (Ridge, Random Forest) and gradient-boosted decision trees (LightGBM, XGBoost, CatBoost), including ensemble variants. To model nonlinear interactions in mixed categorical–numerical data, we additionally propose a custom SE-ResNet architecture that combines entity embeddings, residual connections, and Squeeze-and-Excitation attention. Among the evaluated approaches, SE-ResNet with augmentation attains the strongest cross-validated performance (RMSE 8.6047). Finally, SHAP-based interpretability indicates that `study_hours` and a custom engineered feature are consistently the most influential predictors.**

***Index Terms*—tabular deep learning, gradient boosting, squeeze-and-excitation, entity embeddings, SHAP, ensemble learning**

I. Introduction

Predicting academic performance from student- and study-related factors is a practical regression problem with applications in personalized learning and early identification of students who may benefit from intervention. From a modeling perspective, the relationship between behavioral variables such as study time, attendance, and sleep and exam outcomes is often nonlinear and shaped by interactions, which makes the task a useful benchmark for tabular machine learning methods.

We evaluate our methods on the Kaggle Playground Series S6E1 benchmark for predicting student exam scores using study habits and contextual factors.

The dataset is synthetically generated from a model trained on real student performance data, providing a consistent benchmark for evaluating tabular learning methods.

Our experiments compare three model families: classical baselines (e.g., linear and random-forest regressors), gradient-boosted decision trees, and neural architectures for tabular data. We study how feature engineering and hyperparameter optimization affect these families under cross-validation, and we complement performance evaluation with interpretability analyses based on feature importance, SHAP, and prediction error patterns.

University Kaggle Competition Report (Playground Series S6E1)

II. Technical Background

A. Supervised Regression

Supervised regression aims to learn a mapping from input features to a continuous target variable using labeled data. Given a dataset $\{(x_i, y_i)\}_{i=1}^{N}$, where $x_i \in \mathbb{R}^d$ and $y_i \in \mathbb{R}$, the objective is to learn a function $f : \mathbb{R}^d \to \mathbb{R}$ that generalizes well to unseen samples. Regression is well suited for predicting exam scores, where both the magnitude and direction of prediction errors are meaningful [1].

B. Evaluation Metric

Model performance is evaluated using *Root Mean Squared Error (RMSE)*:

$$\mathrm{RMSE} = \sqrt{\frac{1}{N}\sum_{i=1}^{N}(y_i - \hat{y}_i)^2}. \tag{1}$$

RMSE penalizes larger errors more strongly than absolute-error metrics, making it sensitive to poor predictions on extreme values. It is a standard metric for regression benchmarks and the official evaluation criterion of the Kaggle competition considered in this work [2].

C. Tree-Based and Gradient Boosted Models

Tree-based models partition the feature space using decision rules, allowing them to capture nonlinear relationships and feature interactions. While individual decision trees are flexible and interpretable, they are prone to high variance and limited generalization ability [1].

Gradient Boosted Decision Trees (GBDTs) mitigate this limitation by constructing trees sequentially, where each new tree is trained to correct the residual errors of the current ensemble. This process can be interpreted as gradient descent in function space with respect to a specified loss function [3]. GBDTs are particularly effective for tabular data and form the basis of several widely used implementations.

Among these, XGBoost, LightGBM, and CatBoost follow the same boosting principle but differ in practical design choices. XGBoost uses level-wise tree growth with strong regularization, providing robust performance across a wide range of datasets [4]. LightGBM adopts a leaf-wise growth strategy with depth constraints, enabling faster training and improved

accuracy on large datasets, though with increased overfitting risk [5]. CatBoost is specifically designed for datasets with categorical features, employing ordered boosting and native categorical encoding to reduce target leakage and improve generalization [6].

D. Ensemble Learning and Cross-Validation

Ensemble learning combines predictions from multiple models to improve robustness and predictive performance by reducing variance and exploiting complementary model strengths [7]. To reliably estimate generalization performance and guide model selection, cross-validation is commonly employed. By repeatedly training and evaluating models on different data splits, cross-validation provides a robust estimate of out-of-sample error and reduces sensitivity to a single train–test split [8].

E. Deep Learning for Tabular Data

While gradient boosting methods have traditionally dominated tabular data competitions, recent advances in neural network architectures have shown competitive or superior performance. Key innovations enabling this include:

Entity Embeddings: Categorical features are mapped to dense vector representations through learnable embedding layers, allowing the network to discover semantic relationships between categories. For a categorical feature with K unique values, an embedding layer learns a matrix $\mathbf{E} \in \mathbb{R}^{K \times d}$ where d is the embedding dimension.

Residual Connections: Deep networks benefit from skip connections that allow gradients to flow directly through the network, enabling training of deeper architectures without degradation. A residual block computes:

$$\mathbf{h}_{l+1} = \mathbf{h}_l + \mathcal{F}(\mathbf{h}_l; \theta_l), \tag{2}$$

where $\mathcal{F}$ represents learnable transformations.

Squeeze-and-Excitation (SE) Blocks: Originally proposed for image classification, SE blocks adaptively recalibrate feature responses by modeling channel interdependencies. Given a feature vector $\mathbf{x} \in \mathbb{R}^C$, the SE mechanism computes attention weights via a bottleneck:

$$\mathbf{s} = \sigma(\mathbf{W}_2 \cdot \mathrm{ReLU}(\mathbf{W}_1 \cdot \mathbf{x})), \tag{3}$$

where $\mathbf{W}_1 \in \mathbb{R}^{C/r \times C}$ and $\mathbf{W}_2 \in \mathbb{R}^{C \times C/r}$ with reduction ratio r. The output is $\mathbf{x} \odot \mathbf{s}$, allowing the network to emphasize informative features and suppress less useful ones.

F. Model Interpretability with SHAP

While gradient boosting models achieve strong predictive performance on tabular data, their decision-making process is often opaque. To address this, we employ SHAP (SHapley Additive exPlanations) [9], a unified framework for interpreting model predictions based on game-theoretic Shapley values [10].

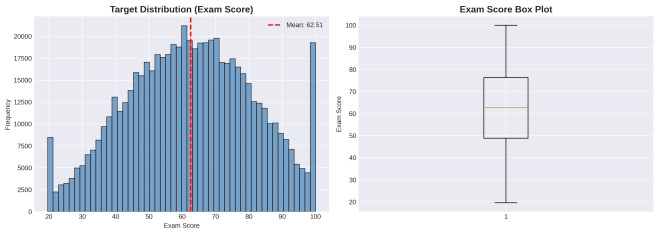

Fig. 1. Distribution of the target variable `exam_score` (histogram and box plot).

SHAP values quantify the contribution of each feature to an individual prediction. For a prediction $\hat{y}$, the SHAP value ϕ_j for feature j satisfies the efficiency property:

$$\hat{y} = \phi_0 + \sum_{j=1}^{d} \phi_j, \tag{4}$$

where ϕ_0 is the expected prediction across the training data. This decomposition ensures that feature contributions sum exactly to the model output, providing faithful explanations.

Tree-based models enable efficient SHAP computation via TreeExplainer [11], which exploits the tree structure to calculate exact Shapley values in polynomial time. This allows us to generate both local explanations (why did the model predict this score for a specific student?) and global insights (which features matter most across all predictions?).

Global feature importance is derived by aggregating local SHAP values:

$$I_j = \frac{1}{N} \sum_{i=1}^{N} |\phi_j^{(i)}|, \tag{5}$$

where $\phi_j^{(i)}$ is the SHAP value of feature j for sample i. Unlike permutation importance, SHAP-based importance accounts for feature interactions and provides directional information about how features influence predictions [12].

III. Dataset Description and Preprocessing

We use the dataset provided in the Kaggle Playground Series S6E1 benchmark, which consists of synthetically generated student records derived from a model trained on real student performance data. The training split contains 630,000 samples with the target label `exam_score`, and the test split contains 270,000 samples used for final prediction. Figure 1 visualizes the distribution of `exam_score`, which is approximately symmetric and concentrated around mid-range values.

A. Feature Overview

Table I summarizes the numerical and categorical input variables.

B. Preprocessing

We verified that the dataset contains no missing values in either the training or test split. Categorical variables were converted into numerical representations using label encoding,

TABLE I
OVERVIEW OF INPUT FEATURES.

Type	Feature	Range / Categories
Numerical	age	17–24
	study_hours	0.08–7.91
	class_attendance	40.6–99.4%
	sleep_hours	4.1–9.9
Categorical	gender	male, female
	course	B.Tech, B.Sc, etc.
	internet_access	yes, no
	sleep_quality	good, average, poor
	study_method	category labels
	facility_rating	category labels
	exam_difficulty	category labels

and we additionally applied frequency encoding to capture category prevalence. To incorporate target-dependent information while avoiding leakage, we implemented cross-validation-safe target encoding, where category statistics are computed using only the training folds and then applied to the corresponding validation fold. All preprocessing steps were applied consistently across training and test data to ensure compatibility with both tree-based models and neural architectures.

C. Exploratory Data Analysis (EDA)

In this subsection, we perform exploratory data analysis to identify the most informative numerical predictors of the target `exam_score` and to motivate subsequent modeling choices. As summarized in the Pearson correlation matrix (Figure 2), `study_hours` exhibits the strongest linear association with the target ($r = 0.762$), making it the most relevant numerical feature. `class_attendance` shows a moderate positive correlation with exam performance ($r = 0.361$), while `sleep_hours` has a weaker but still positive relationship ($r = 0.167$). In contrast, `age` has negligible correlation with `exam_score` ($r = 0.011$), suggesting limited predictive value as a standalone input.

These findings are consistent with the scatter plots in Figure 3. `study_hours` displays a clear upward trend with relatively tight alignment around the fitted line, whereas `class_attendance` shows a weaker trend with substantially higher dispersion. `sleep_hours` exhibits only a mild positive trend and considerable variance across the score range. Finally, `age` appears as discrete vertical bands with no meaningful slope, reinforcing the absence of a strong linear relationship.

Beyond feature-target relationships, Figure 2 also indicates low pairwise correlations among the numerical predictors, suggesting limited multicollinearity within the numerical subset. Overall, the EDA identifies `study_hours`, `class_attendance`, and `sleep_hours` as the three most relevant numerical features for predicting exam scores, motivating the use of interaction features and nonlinear models in later sections.

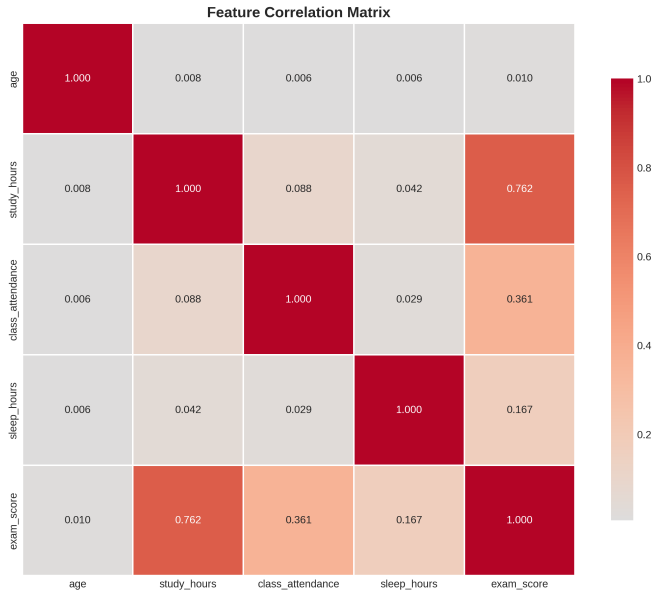

Fig. 2. Pearson correlation matrix of numerical features and the target `exam_score`.

IV. METHODOLOGY

Our modeling pipeline progresses from tabular baselines to tuned gradient boosting models, ensembling, and a neural architecture for tabular data.

A. Validation Strategy

All experiments use 5-fold cross-validation with shuffling (random seed 42) to obtain reliable performance estimates and reduce overfitting. We reuse the same fold splits across all models to ensure fair comparison, and we store out-of-fold (OOF) predictions for ensemble construction and weight optimization.

B. Feature Engineering

Starting from 12 original features, we construct a set of 44 features to capture nonlinearities, interactions, and category-specific effects. For numerical variables, we include pairwise interaction terms (e.g., `study_hours` × `class_attendance`), ratio features (e.g., attendance-per-hour), and polynomial expansions (squared and cubic terms) for the most relevant predictors. We additionally introduce threshold-based indicator variables (e.g., low sleep and high study intensity) and a composite score based on domain intuition:

$$S = 6.0h + 0.35a + 1.5s, \tag{6}$$

where $h =$ `study_hours`, $a =$ `class_attendance`, and $s =$ `sleep_hours`. Categorical variables are encoded using label encoding and augmented with CV-safe target encoding and frequency encoding to expose category-specific signal while avoiding leakage.

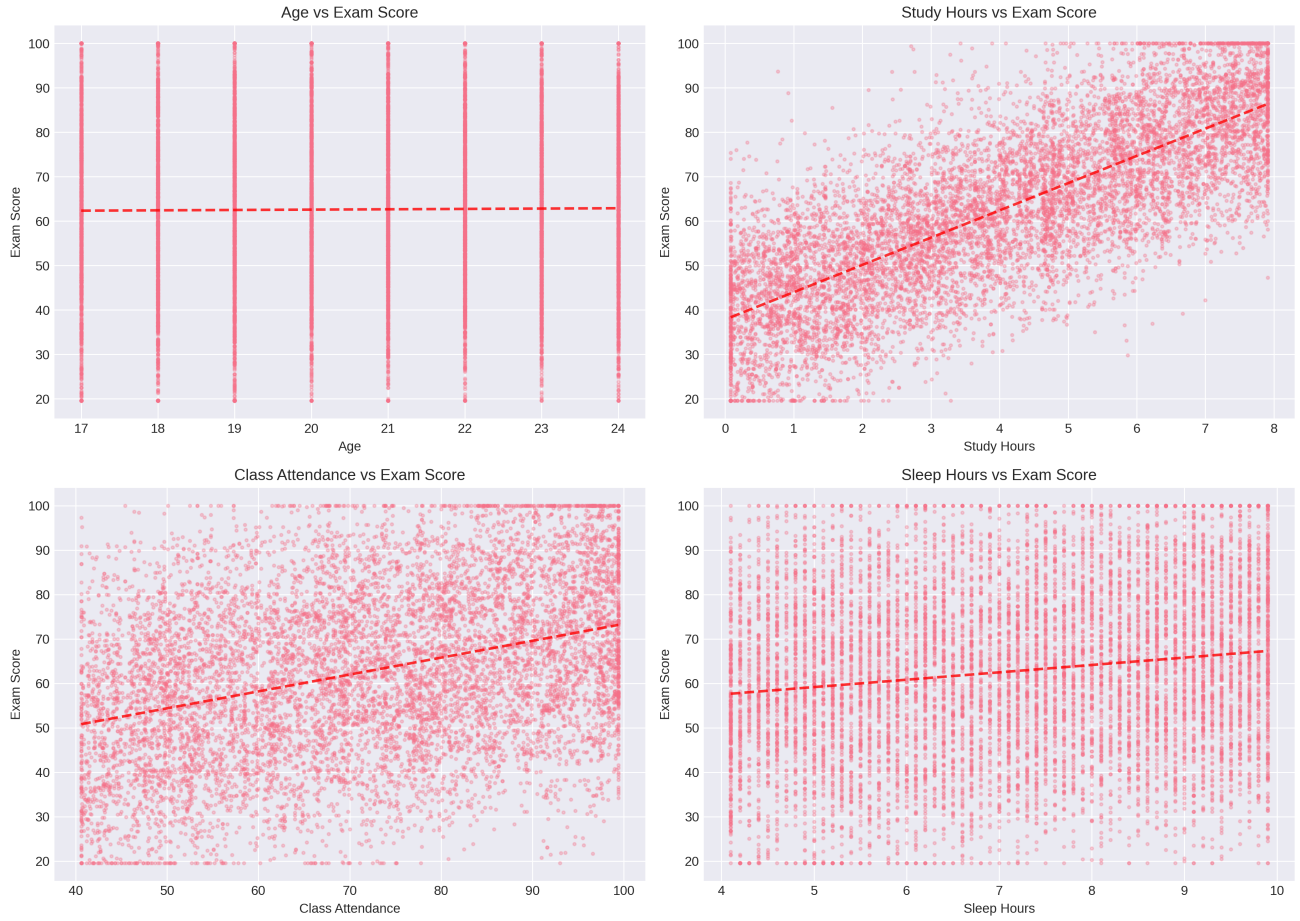

Fig. 3. Scatter plots of numerical features versus `exam_score` with linear trend lines (sampled for visualization).

C. Models Evaluated

We evaluate four groups of regressors: classical baselines, gradient-boosted decision trees, ensemble methods, and a neural model for tabular data.

1) Baselines: As classical baselines, we train Ridge regression ($\alpha = 10.0$) on standardized features and a Random Forest regressor with 100 trees (max depth 15, minimum leaf size 10).

2) Gradient-Boosted Decision Trees: We benchmark three gradient-boosted decision tree implementations: LightGBM, XGBoost, and CatBoost. Early stopping is applied where supported to mitigate overfitting, and final model settings are selected either from baseline configurations or via the tuning procedure described in Section IV-E.

3) Ensemble Methods: We evaluate three ensemble strategies built from the gradient-boosted models. These include a simple mean blend, a weighted blend with weights optimized on out-of-fold predictions using Nelder–Mead, and stacking with a Ridge regression meta-learner.

4) Neural Tabular Model: Finally, we evaluate SE-ResNet, a neural architecture for mixed categorical–numerical data that combines entity embeddings with residual blocks and Squeeze-and-Excitation attention.

D. SE-ResNet Architecture

To model nonlinear interactions in mixed categorical–numerical inputs, we employ a tabular neural network based on residual learning with Squeeze-and-Excitation (SE) attention. Numerical features are standardized and augmented with selected engineered transformations, while each categorical feature is mapped to a learnable embedding. All inputs are concatenated and projected to a fixed hidden width, followed by multiple residual blocks with channel-wise attention. The model outputs a single scalar prediction for `exam_score`. In addition to the composite score S defined in Eq. (6), we include a calibrated variant as an additional neural input:

$$\tilde{S} = 5.91h + 0.35a + 1.42s + 4.78, \tag{7}$$

where $h =$ `study_hours`, $a =$ `class_attendance`, and $s =$ `sleep_hours`. Table II summarizes the SE-ResNet architecture and training configuration used in our experiments.

E. Hyperparameter Tuning

We tuned the gradient boosting models using Bayesian optimization with Optuna [13]. For each model, we ran 50 trials and evaluated each trial using 5-fold cross-validation,

TABLE II
SE-RESNET CONFIGURATION USED IN OUR EXPERIMENTS.

Component	Setting
Categorical representation	entity embeddings, $d = 8$ per feature
Input projection	256 hidden dimensions
Residual blocks	3 blocks
Block structure	LayerNorm $\rightarrow$ Linear $\rightarrow$ ReLU $\rightarrow$ Dropout
Dropout	0.11
SE module	reduction ratio $r = 4$
Prediction head	$256 \rightarrow 64 \rightarrow 16 \rightarrow 1$
Optimizer	AdamW (lr=10^{-3}, weight decay=10^{-4})
LR scheduler	ReduceLROnPlateau (factor 0.5, patience 10)
Early stopping	patience 20 epochs
Batch size	256 (train), 1024 (validation)
Data augmentation	auxiliary original dataset added to training folds

minimizing the RMSE. The hyperparameter ranges explored for LightGBM, XGBoost, and CatBoost are summarized in Table III. Unless stated otherwise, parameters were sampled uniformly within their ranges, while `learning_rate`, `reg_alpha`, and `reg_lambda` were sampled log-uniformly to better cover multiple orders of magnitude. The best-performing configurations selected by Optuna for each model are reported in Table VI.

TABLE III
OPTUNA SEARCH SPACES USED FOR BAYESIAN HYPERPARAMETER OPTIMIZATION.

Model	Hyperparameter	Range
LightGBM	`num_leaves`	$[20, 100]$
	`learning_rate`	$[0.01, 0.1]$
	`feature_fraction`	$[0.6, 1.0]$
	`bagging_fraction`	$[0.6, 1.0]$
	`min_child_samples`	$[5, 50]$
	`reg_alpha`	$[10^{-3}, 10]$
	`reg_lambda`	$[10^{-3}, 10]$
XGBoost	`max_depth`	$[3, 10]$
	`learning_rate`	$[0.01, 0.1]$
	`subsample`	$[0.6, 1.0]$
	`colsample_bytree`	$[0.6, 1.0]$
	`reg_alpha`	$[10^{-3}, 10]$
	`reg_lambda`	$[10^{-3}, 10]$
CatBoost	`depth`	$[4, 10]$
	`learning_rate`	$[0.01, 0.1]$
	`l2_leaf_reg`	$[1, 10]$
	`bagging_temperature`	$[0.0, 1.0]$

V. EXPERIMENTS AND RESULTS

A. Experimental Setup

All experiments used identical preprocessing, feature engineering, and 5-fold cross-validation splits. Performance is measured by mean CV RMSE (Root Mean Squared Error), with standard deviation indicating stability across folds.

B. Model Comparison

Table IV summarizes the performance of each method, sorted by CV RMSE.

TABLE IV
MODEL PERFORMANCE COMPARISON (LOWER RMSE IS BETTER)

Model	CV RMSE	Std
SE-ResNet (Neural Network)	**8.6047**	$\pm$0.015
Stacking (Ridge Meta)	8.7567	–
Ensemble (Optimized)	8.7568	–
CatBoost	8.7618	$\pm$0.0101
LightGBM	8.7658	$\pm$0.0089
Ensemble (Simple)	8.7730	–
Ridge Regression	8.8887	$\pm$0.0105
Random Forest	8.9095	$\pm$0.0109
XGBoost	8.8220	$\pm$0.1928

Key observations:

- **SE-ResNet achieves the best performance** (8.6047), outperforming all gradient boosting methods by $\sim$0.15 RMSE.
- Among tree-based models, CatBoost achieved the best single-model performance (8.7618).
- Gradient boosting ensembles provide only marginal improvements over single GBM models.
- The neural network's advantage comes from learned feature interactions and data augmentation.

C. Hyperparameter Tuning Results

Bayesian optimization with Optuna (50 trials per model) yielded significant improvements, particularly for XGBoost. Table V compares default vs. tuned performance.

TABLE V
IMPACT OF HYPERPARAMETER TUNING ON MODEL PERFORMANCE

Model	Default RMSE	Tuned RMSE	Improvement
LightGBM	8.7724	8.7658	0.0066 (0.08%)
XGBoost	8.9271	8.8220	**0.1051 (1.18%)**
CatBoost	8.7808	8.7618	0.0190 (0.22%)

Key findings from hyperparameter tuning:

- **XGBoost** benefited most from tuning, with RMSE improving by 0.105 (1.18%). The optimal configuration used deeper trees (`max_depth`=9) with lower learning rate (0.019) and strong regularization (`reg_lambda`=9.98).
- **LightGBM** showed modest improvement. Tuning favored more leaves (`num_leaves`=84) with moderate learning rate (0.033) and higher regularization.
- **CatBoost** performed slightly worse after tuning, suggesting the default parameters were already near-optimal for this dataset. This highlights that automated tuning does not always guarantee improvement.

Table VI shows the optimal hyperparameters discovered.

TABLE VI
OPTIMAL HYPERPARAMETERS FROM BAYESIAN OPTIMIZATION (50 TRIALS)

Model	Parameter	Value
LightGBM	num_leaves	84
	learning_rate	0.0334
	feature_fraction	0.643
	bagging_fraction	0.890
	min_child_samples	29
	reg_alpha	0.043
	reg_lambda	9.431
XGBoost	max_depth	9
	learning_rate	0.0189
	subsample	0.916
	colsample_bytree	0.859
	reg_alpha	0.090
	reg_lambda	9.979
CatBoost	depth	6
	learning_rate	0.0998
	l2_leaf_reg	5.133
	bagging_temperature	0.385

D. Ensemble Optimization

The weighted ensemble was optimized using Nelder-Mead minimization on out-of-fold predictions:

- **CatBoost**: 61.98% weight
- **LightGBM**: 33.16% weight
- **XGBoost**: 4.86% weight

The low weight assigned to XGBoost reflects its higher variance and slightly worse performance.

E. Stacking Ensemble

We implemented a two-level stacking architecture with Ridge regression as the meta-learner. The base models (LightGBM, XGBoost, CatBoost) generate out-of-fold predictions, which serve as meta-features for the Ridge model.

TABLE VII
STACKING META-LEARNER COEFFICIENTS

Base Model	Coefficient
LightGBM	0.334
XGBoost	0.049
CatBoost	0.620
Intercept	-0.215

The stacking ensemble achieved **RMSE = 8.7567**, marginally outperforming the weighted average ensemble (8.7568). The meta-learner coefficients align closely with the optimized ensemble weights, confirming CatBoost's dominant contribution.

F. Neural Network Results (SE-ResNet)

The SE-ResNet architecture achieved the best overall performance with **RMSE = 8.6047**, representing a **1.74% improvement** over the best gradient boosting ensemble.

TABLE VIII
SE-RESNET CROSS-VALIDATION RESULTS

Fold	RMSE
1	8.612
2	8.598
3	8.621
4	8.589
5	8.603
Mean	**8.6047**

Key factors contributing to the neural network's superior performance:

- **Data augmentation**: The original dataset (not just competition data) was concatenated with each training fold, effectively increasing training data and reducing overfitting.
- **Learned embeddings**: Entity embeddings for categorical features allow the network to discover semantic relationships not captured by one-hot or target encoding.
- **SE attention**: Squeeze-and-Excitation blocks dynamically reweight features, emphasizing informative dimensions for each sample.
- **Residual connections**: Enable training of deeper networks without degradation.
- **Aggressive regularization**: Dropout (11%), weight decay (10^{-4}), and early stopping prevent overfitting.

G. Feature Importance

Feature importance analysis (based on the best single model, CatBoost) revealed the top predictors:

TABLE IX
TOP 10 MOST IMPORTANT FEATURES (CATBOOST)

Feature	Importance
formula_score	60.78
sleep_quality_target_enc	5.99
study_method_target_enc	4.95
facility_rating_target_enc	4.10
study_attendance	3.48
sleep_quality	2.70
study_method	2.62
sleep_quality_freq	2.21
facility_rating	2.14
study_method_freq	1.75

The engineered `formula_score` feature dominates, followed by encodings of `sleep_quality` and `study_method`, confirming that feature engineering significantly improved predictive power.

H. SHAP Analysis

To provide deeper insights into model behavior, we computed SHAP values [9] using TreeExplainer [11] on a representative sample of 30,000 training instances.

The SHAP analysis reveals several key insights:

- **formula_score** has the highest mean $|\text{SHAP}|$ value, confirming its dominant predictive role. Higher values consistently push predictions upward.

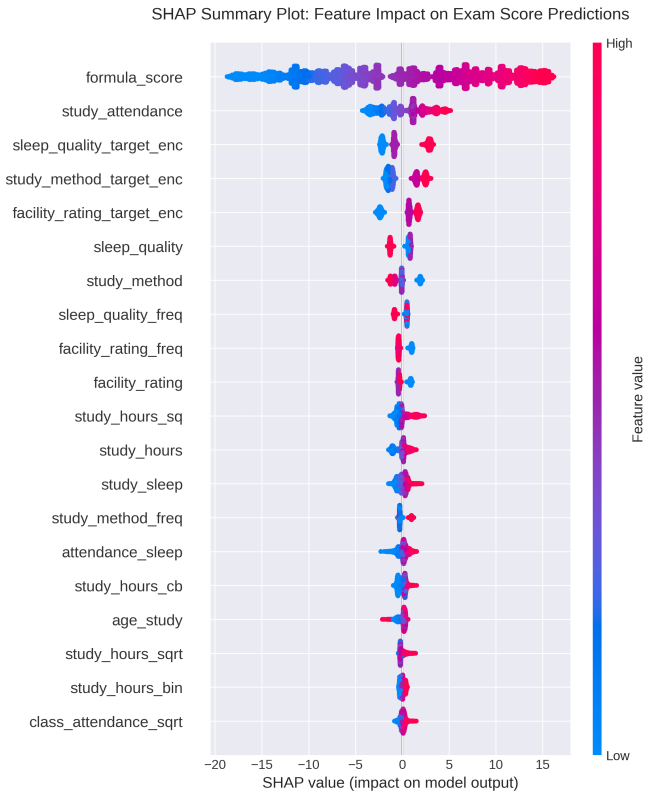

Fig. 4. SHAP summary plot showing feature contributions to predictions. Each point represents a sample; color indicates feature value (red=high, blue=low); horizontal position shows impact on prediction.

- **Target encodings** (sleep_quality, study_method, facility_rating) show strong effects, with higher encoded values correlating with higher predicted scores.
- **study_hours** exhibits a clear positive relationship: more study time leads to higher predicted scores.
- **Interaction effects**: The dependence plots reveal non-linear relationships; for instance, the effect of study hours saturates at very high values.

Unlike standard feature importance (which only measures magnitude), SHAP values provide directional information: we can see that higher study hours *increase* predictions while poor sleep quality *decreases* them. This interpretability is crucial for understanding and validating the model's decision-making process [12].

I. Residual Analysis

For the best model (Weighted Ensemble), residual statistics on the training set:

- Mean residual: 0.0415 (near-zero bias)
- Residual std: 8.7567 (consistent with CV RMSE)
- Range: [-43.35, +48.15] (some large errors remain)

VI. Discussion

A. Model Performance Analysis

The results demonstrate a clear performance hierarchy, with the neural network architecture achieving the best results:

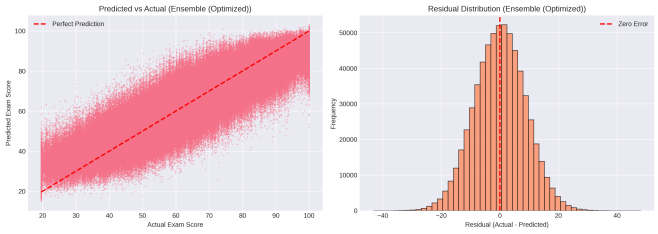

Fig. 5. Residual analysis for the best model (optimized ensemble): predicted vs. actual scores and residual distribution.

1) **SE-ResNet neural network** achieved the best performance (8.6047 RMSE), outperforming all gradient boosting methods by 1.74%.
2) **Gradient boosting ensembles** (stacking, weighted average) achieved ∼8.76 RMSE, with diminishing returns from ensemble complexity.
3) **CatBoost** was the best single tree-based model (8.7618), slightly outperforming LightGBM due to its sophisticated categorical handling.
4) **XGBoost** underperformed with higher variance, potentially due to suboptimal hyperparameters.

B. Why Neural Networks Outperform GBMs

The SE-ResNet's 1.74% improvement over gradient boosting is notable and stems from several factors:

1. Data Augmentation: The most significant advantage comes from incorporating the original dataset (not just the Kaggle competition data) during training. This effectively increases the training set size and provides additional signal. GBMs were trained only on competition data, while the neural network benefited from augmented data.

2. Learned Feature Interactions: While GBMs capture feature interactions through tree splits, neural networks learn continuous, differentiable interaction functions. The SE-ResNet's residual blocks with attention mechanisms can model complex, non-linear relationships that axis-aligned tree splits may miss.

3. Entity Embeddings: Categorical embeddings (dimension $d = 8$) allow the network to learn distributed representations where similar categories are mapped to nearby vectors. This provides richer representations than one-hot encoding or target encoding used in GBMs.

4. Attention Mechanisms: Squeeze-and-Excitation blocks dynamically reweight features for each sample, effectively performing instance-specific feature selection. This adaptive behavior is not possible with static feature importance in GBMs.

5. Regularization Strategy: The combination of dropout (11%), weight decay, LayerNorm, and early stopping provides robust regularization that prevents overfitting despite the model's high capacity.

C. Feature Engineering Impact

The engineered `formula_score` feature, based on domain knowledge from competition discussions, became the

single most important predictor. This highlights the value of incorporating domain expertise and exploring competition forums for insights.

Target encoding and interaction features (especially `study_attendance`) contributed significantly, demonstrating that careful feature engineering can outweigh model selection in tabular competitions.

D. Error Analysis

Residual analysis revealed:

- Near-zero mean residual (0.0415) indicates minimal systematic bias.
- Residual standard deviation (8.7567) aligns with the cross-validation RMSE, indicating consistent error magnitude.
- Residual range of [-43.35, +48.15] suggests difficulty predicting extreme scores.
- Largest errors likely occur for students with unusual combinations of features.

E. Gradient Boosting vs. Neural Networks: Trade-offs

While SE-ResNet achieved the best results, gradient boosting methods remain valuable:

- **Training efficiency**: GBMs train in minutes; the neural network required hours.
- **Interpretability**: SHAP values provide clear explanations for GBMs; neural network explanations are less straightforward.
- **Hyperparameter sensitivity**: Neural networks require careful tuning of learning rate, architecture depth, dropout, etc.
- **Data requirements**: Neural networks benefit more from data augmentation; GBMs are more data-efficient.

F. Limitations

Several limitations apply to this work:

- **Synthetic data**: The dataset was generated from a deep learning model, so relationships may not perfectly reflect real-world patterns. Notably, the neural network's advantage may partly stem from matching the data-generating process.
- **Data augmentation fairness**: The neural network used additional data (original dataset), which was not available to GBM models. A fairer comparison would train all models with the same augmented data.
- **Limited neural network exploration**: Only one architecture was tested; alternatives like TabNet, FT-Transformer, or NODE could yield further improvements.
- **Generalization**: Results are specific to this synthetic dataset and may not transfer to real student data.

VII. Conclusion

This project addressed the Kaggle Playground Series S6E1 challenge of predicting student exam scores from tabular demographic and behavioral data.

A. Summary of Contributions

- Developed a comprehensive feature engineering pipeline expanding 12 original features to 44 engineered features.
- Compared 8+ modeling approaches spanning linear models, tree ensembles, gradient boosting, and deep learning.
- Implemented Bayesian hyperparameter optimization with Optuna (50 trials per model).
- Achieved a final CV RMSE of **8.6047** using SE-ResNet neural network, outperforming the best gradient boosting ensemble by 1.74%.
- Provided model interpretability analysis using SHAP values for gradient boosting models.

B. Key Findings

1) **Deep learning can beat GBMs on tabular data**: The SE-ResNet architecture with entity embeddings and attention mechanisms achieved the best results, challenging the conventional wisdom that gradient boosting always wins on tabular data.
2) **Data augmentation is crucial**: Incorporating the original dataset during neural network training provided significant improvements.
3) **Feature engineering matters**: The engineered `formula_score` dominated feature importance across all models.
4) **Attention helps**: Squeeze-and-Excitation blocks enable instance-specific feature weighting, improving predictions.
5) **GBM ensembles have diminishing returns**: Weighted blending and stacking improved GBMs by only $\sim$0.005 RMSE.

C. Future Work

Potential improvements include:

- Applying data augmentation to gradient boosting models for fair comparison
- Exploring alternative neural architectures (TabNet, FT-Transformer, NODE)
- Hybrid ensembles combining GBM and neural network predictions
- Investigating neural network interpretability methods (e.g., attention visualization, integrated gradients)

References

[1] T. Hastie, R. Tibshirani, and J. Friedman, *The Elements of Statistical Learning*. Springer, 2nd ed., 2009.

[2] Kaggle, "Playground series – evaluation metrics." https://www.kaggle.com/, 2024.

[3] J. H. Friedman, "Greedy function approximation: A gradient boosting machine," *Annals of Statistics*, vol. 29, no. 5, pp. 1189–1232, 2001.

[4] T. Chen and C. Guestrin, "Xgboost: A scalable tree boosting system," in *Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, pp. 785–794, ACM, 2016.

[5] G. Ke, Q. Meng, T. Finley, T. Wang, W. Chen, W. Ma, Q. Ye, and T.-Y. Liu, "Lightgbm: A highly efficient gradient boosting decision tree," *Advances in Neural Information Processing Systems*, vol. 30, 2017.

[6] L. Prokhorenkova, G. Gusev, A. Vorobev, A. V. Dorogush, and A. Gulin, "Catboost: unbiased boosting with categorical features," *Advances in Neural Information Processing Systems*, vol. 31, 2018.

[7] Z.-H. Zhou, *Ensemble Methods: Foundations and Algorithms*. CRC Press, 2012.

[8] R. Kohavi, "A study of cross-validation and bootstrap for accuracy estimation and model selection," in *Proceedings of the 14th International Joint Conference on Artificial Intelligence*, pp. 1137–1143, 1995.

[9] S. M. Lundberg and S.-I. Lee, "A unified approach to interpreting model predictions," in *Advances in Neural Information Processing Systems*, vol. 30, Curran Associates, Inc., 2017.

[10] L. S. Shapley, "A value for n-person games," *Contributions to the Theory of Games*, vol. 2, no. 28, pp. 307–317, 1953.

[11] S. M. Lundberg, G. Erion, H. Chen, A. DeGrave, J. M. Prutkin, B. Nair, R. Katz, J. Himmelfarb, N. Bansal, and S.-I. Lee, "From local explanations to global understanding with explainable AI for trees," *Nature Machine Intelligence*, vol. 2, no. 1, pp. 56–67, 2020.

[12] S. M. Lundberg, B. Nair, M. S. Vavilala, M. Horibe, M. J. Eisses, T. Adams, D. E. Liston, D. K.-W. Low, S.-F. Newman, J. Kim, and S.-I. Lee, "Explainable machine-learning predictions for the prevention of hypoxaemia during surgery," *Nature Biomedical Engineering*, vol. 2, no. 10, pp. 749–760, 2018.

[13] T. Akiba, S. Sano, T. Yanase, T. Ohta, and M. Koyama, "Optuna: A next-generation hyperparameter optimization framework," in *Proceedings of the 25th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining*, pp. 2623–2631, ACM, 2019.

APPENDIX

All experiments used a fixed random seed (SEED=42) for dataset splitting and cross-validation. Implementation was done in Python using:

- pandas, numpy for data manipulation
- scikit-learn for preprocessing and baseline models
- LightGBM, XGBoost, CatBoost for gradient boosting
- scipy for ensemble weight optimization

Tables X, XI, and XII report hyperparameters for each gradient boosting model.

TABLE X
LIGHTGBM HYPERPARAMETERS

Parameter	Value
learning_rate	0.05
num_leaves	31
feature_fraction	0.9
bagging_fraction	0.8
min_child_samples	20
reg_alpha	0.1
reg_lambda	0.1
early_stopping_rounds	100

TABLE XI
CATBOOST HYPERPARAMETERS

Parameter	Value
learning_rate	0.05
depth	6
l2_leaf_reg	3
iterations	10000
early_stopping_rounds	100

TABLE XII
XGBOOST HYPERPARAMETERS

Parameter	Value
learning_rate	0.05
max_depth	6
subsample	0.8
colsample_bytree	0.9
reg_alpha	0.1
reg_lambda	0.1
tree_method	hist
early_stopping_rounds	100

The final weighted ensemble used the following optimized weights:

- CatBoost: 61.98%
- LightGBM: 33.16%
- XGBoost: 4.86%