Student Performance Analytics

15.072: Advanced Analytics Edge

Fall 2024

Nidhish Nerur, Brimar Ólafsson, James Pinter, Angel Xie, Yuki Yu

1 Problem Motivation

Education is crucial to shaping future opportunities, making it vital to understand factors influencing student performance. Our project focuses on predicting academic success for the early identification of struggling students. With these predictions, targeted interventions, such as tutoring or counseling, can improve outcomes and help students reach their potential. By identifying key factors for academic success and challenges, we aim to guide resource allocation and educational policies. Ultimately, this project seeks to provide data-driven recommendations to foster equitable and successful learning environments.

Historically, educators have utilized descriptive statistics and regression analysis to examine factors affecting student performance. These traditional methods effectively identify correlations between variables such as attendance and socioeconomic status with academic outcomes, but fail to capture non-linear relationships. A recent study on forecasting student academic performance demonstrated that machine learning algorithms, including Naïve Bayes, K-Nearest Neighbor, Support Vector Machines, and Neural Networks, have significantly improved prediction accuracy when applied to educational datasets, outperforming conventional models (Dawar et al., 2024). Our project seeks to leverage similar machine learning methodologies to uncover deeper insights into the key determinants of student academic performance.

2 Dataset Description

Kaggle Dataset We have a dataset from Kaggle, comprising 6,607 student records with 20 features. These features encompass various aspects of student life, including classroom discipline (e.g., Hours_Studied, Attendance), home environment (e.g., Parental_Involvement, Family_Income, Internet_Access), and extracurricular activities (e.g., Extracurricular_Activities, Physical_Activity). The target variable for our predictive models is the student's final exam score, providing a quantitative measure of academic performance.
Dataset Source: https://www.kaggle.com/datasets/lainguyn123/student-performance-factors

2.1 Data Augmentation

Synthetic Data Generation Given the relatively small size of our combined dataset (6,607 observations), we supplemented it with synthetic data. We utilized Synthetic Minority Oversampling Technique (SMOTE) to duplicate instances of minority classes in classification. Additionally, we utilized Gaussian Mixture Models to estimate numeric features in a way that resembled the underlying distribution of data. Our augmentation approach aims to provide a more diverse and balanced dataset with additional observations, potentially enhancing our models' predictive capabilities. However, we carefully evaluate the benefits of synthetic data against the risk of overfitting. The results for the augmented models are found in the Appendix.

2.2 Data Processing

Our preprocessing approach follows the data science pipeline, encompassing exploratory data analysis, cleaning, and feature engineering. We view the correlation matrix between numeric features and do not see significant risk of multi-collinearity:

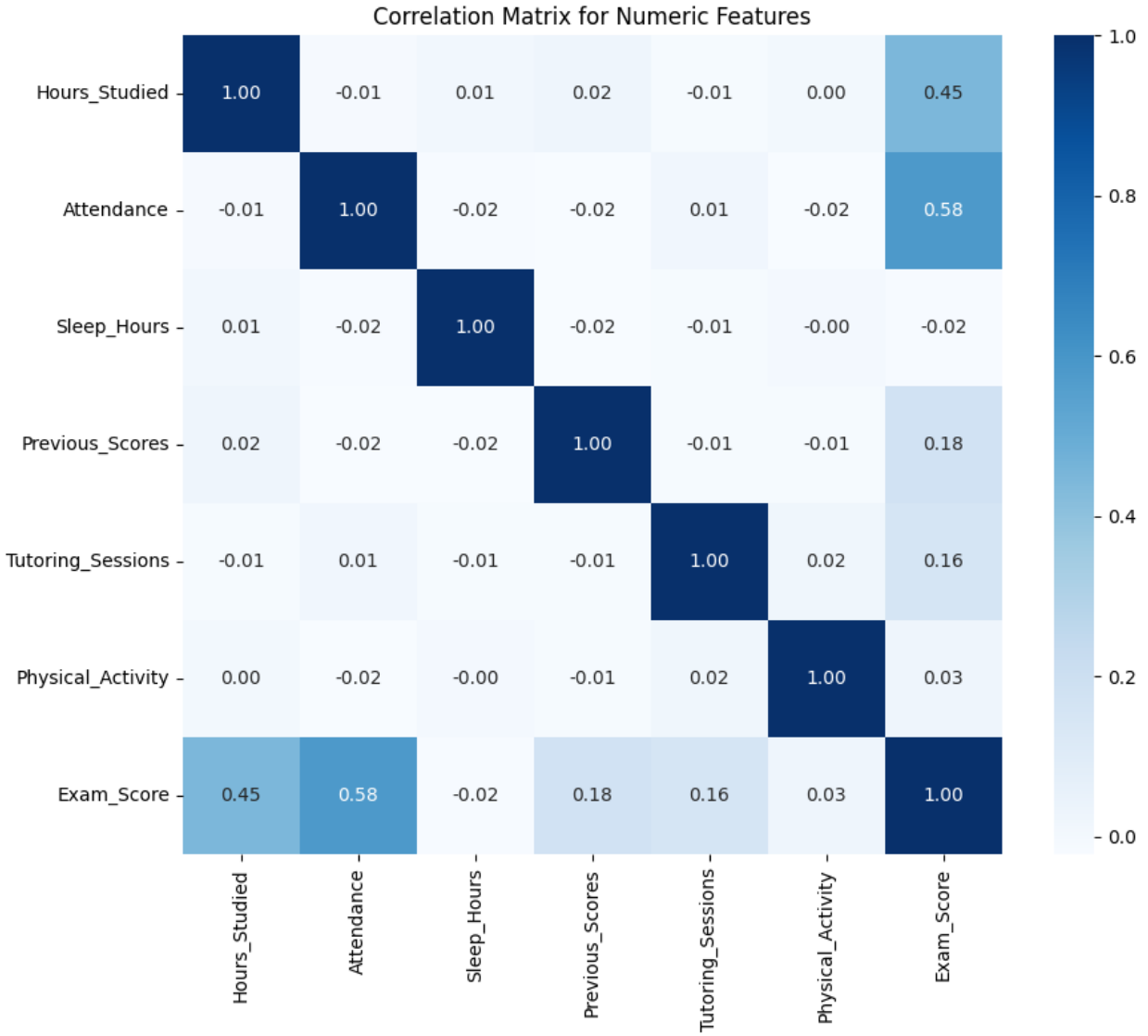

Figure 1: Correlation Matrix

Then, we found linear relationships between attendance rate and exam score as well as the hours studied and exam score, which are shown in the appendix. We created new interactions between hours studied and attendance, learning engagement, home support levels, study to sleep ratio, and balance of physical activity with sleep. Finally, we handled missing values, encoded categorical variables, and normalized numeric features to use machine learning algorithms.

3 Regression Task

For the regression task, we tested linear regression models, decision trees, and ensemble methods such as random forest, gradient boosting, and neural networks. We built baseline models and highlight the results:

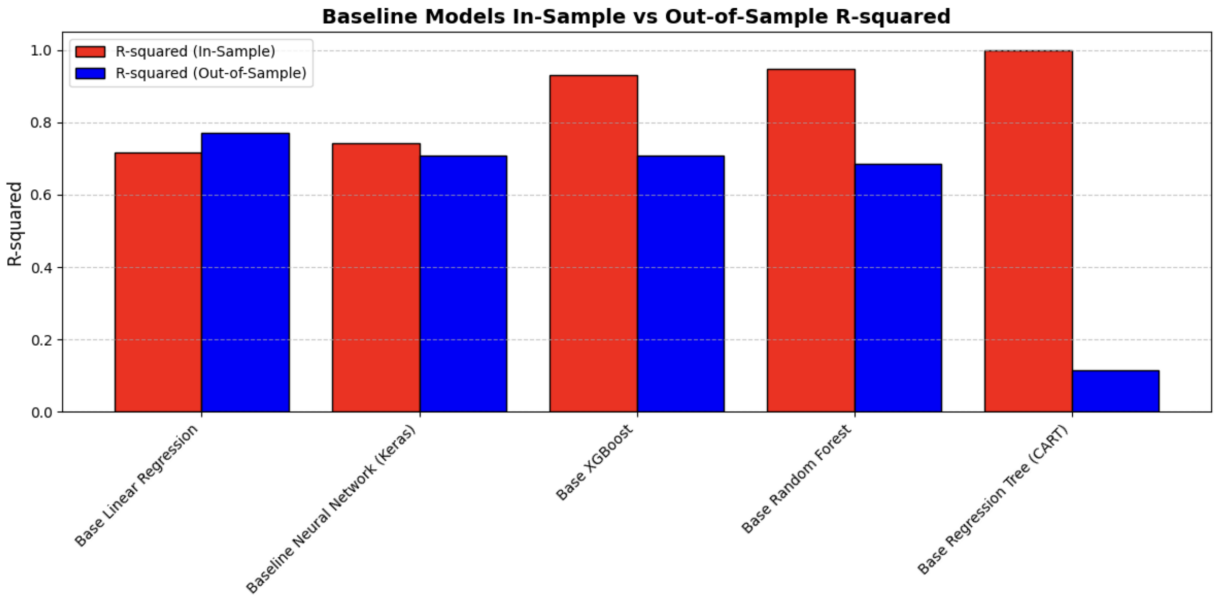

Figure 2: Baseline Regression Plots

Notice the XGBoost, Random Forest, and CART tree appear to be heavily overfitting to the training set, given their in-sample R-squared scores are much greater than the out-of-sample R-squared values. Consequently, we perform hyperparameter tuning across all our models and add the Lasso, Ridge, ElasticNet models. Furthermore, we used Bayesian Optimization to find optimal hyperparameters by balancing exploration of new values and refinement of promising ones, reducing the need for exhaustive searches. The table below shows the performance metrics of tuned models that performed better than their baselines.

Table I: Tuned Model Performance Metrics

Model	R^2 **(In)**	**OOS** R^2	**MSE (In)**	**OOS MSE**	**MAE (In)**	**OOS MAE**
ElasticNet	0.7169	0.7716	4.3529	3.23172	0.5011	0.4513
Ridge Regression	0.7171	0.7711	4.3506	3.2354	0.4970	0.4483
XGBoost	0.7739	0.7500	3.4767	3.5325	0.6280	0.6767
Lasso Regression	0.6827	0.7436	4.8793	3.6240	0.8317	0.7573
CART	0.6478	0.6353	5.1541	5.1541	0.9473	1.2378

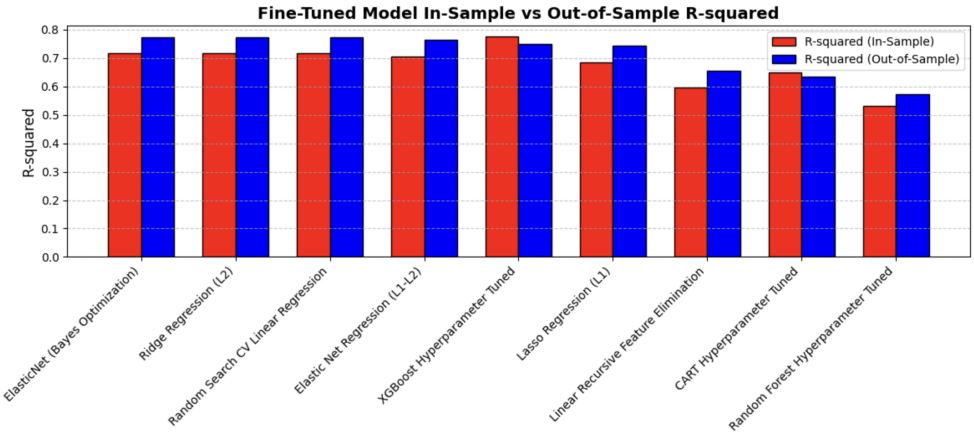

Figure 3: Hyperparameter Tuning Results for Different Models

Finally, here are our results from ensembling the top regression models:

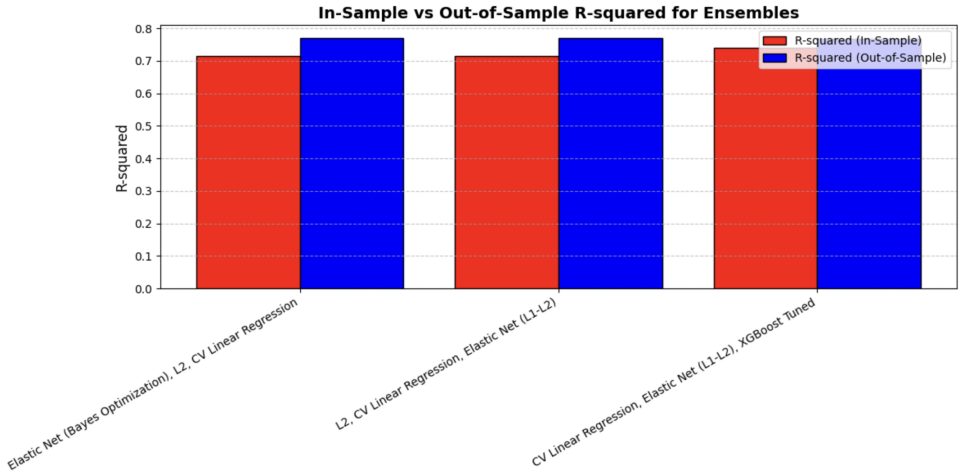

Figure 4: Ensemble of Top Regression Models

3.0.1 Lasso Regression

We used `RandomizedSearchCV` to tune Lasso's α hyperparameter with 5-fold cross-validation. The best α was 0.1, resulting in a mean squared error (MSE) of 3.624 and an R^2 score of 0.743. Bayesian Optimization was not used, as Lasso showed the largest performance drop compared to Ridge and ElasticNet, likely due to its stricter sparsity constraints. Our best lasso model has 14 non-zero features, making it a bit more interpretable than models including all the features. From a business perspective, we can directly interpret the statistical significance of features and achieves comparable performance to the other models tested.

3.0.2 Ridge Regression

We used Bayesian Optimization to tune the α (regularization strength) hyperparameter for Ridge regression. The best α found was approximately 3.576, achieving a cross-validation score of -4.401, a test MSE of 3.234, and an R^2 score of 0.771. Ridge is less interpretable than lasso given it does not zero out any of the coefficients, but we can still intuitively explain the relationship between the features and the final exam score outcome.

3.0.3 Elastic Net Regression

We used Bayesian Optimization to tune the α (regularization strength) and l1_ratio (balance between Lasso and Ridge) hyperparameters for ElasticNet. The optimization identified the best parameters as $\alpha \approx 0.0039$ and l1_ratio ≈ 0.5136, achieving a cross-validation score of -4.399, a test MSE of 3.232, and an R^2 score of 0.772. Elastic Net utilizes a combination of L1 and L2 regularization, so it can zero some terms while retaining others. This model had strong performance with out of sample R-squared of around 76%, close to our top three models, and would make a practical choice in a business setting. Given this is our highest performing model, we show the most important features selected:

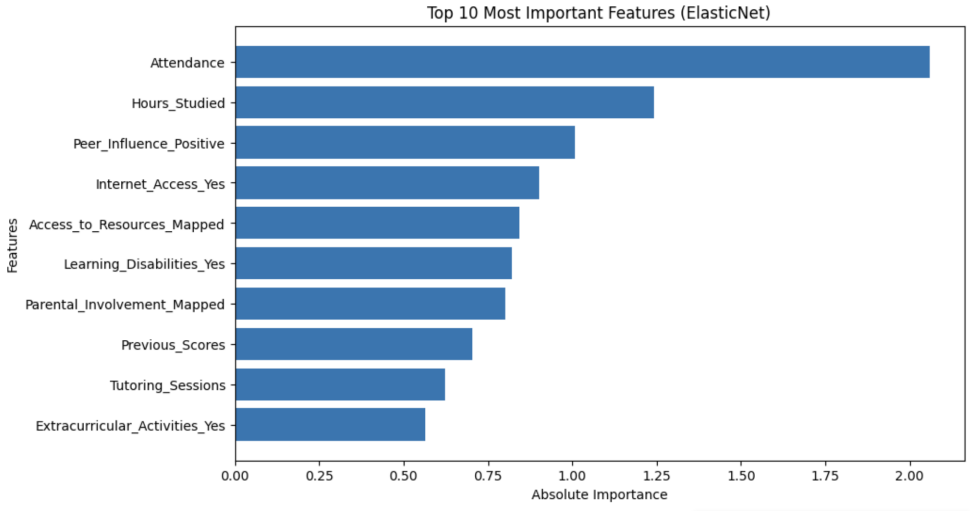

Figure 5: Most Important Features ElasticNet

Logically, attendance rates, number of hours studied, and positive peer influence make sense in determining a student's exam score. It appears that attendance is the most notable feature by quite a large margin, and we expect students who attend class more often are likely understanding the course material better than those who do not. ElasticNet balances interpretability of coefficients with the strongest performance.

3.0.4 CART

We used Bayesian Optimization to tune the hyperparameters for the Decision Tree (CART) model. The optimization process identified the best parameters as:

- `max_depth`: 21
- `min_samples_split`: 2
- `min_samples_leaf`: 16

These parameters achieved a cross-validation score of -6.988, a test Mean Squared Error (MSE) of 5.810, and an R^2 score of 0.589. CART is highly interpretable, given we can map precisely how to arrive at a particular exam score prediction. However, it had the second lowest performance, with an out of sample R-squared of 63.54

3.0.5 Random Forest

The Random Forest Regressor performed close to the XGBoost model without hyperparameter tuning, but these models were still worse than our linear models overall. After hyperparameter tuning, Random Forest had the lowest out of sample R-squared score below 60%, which is surprising. Since our dataset has primarily linear relationships with the target, it seems the tuned CART tree outperformed random forest. We experimented with the number of trees, maximum number of features, and max depth of the tree, yet it still had the worst performance. Additionally, this model is not interpretable as it aggregates decision trees.

3.0.6 XGBoost

The best Bayesian Optimization with Hyperopt XGBoost hyperparameters include:

- `max_depth`: 3,
- `gamma`: 1.47,
- `reg_alpha`: 40,
- `reg_lambda`: 0.96,
- `colsample_bytree`: 0.97,
- `min_child_weight`: 3.

These parameters achieved a test MSE of 3.474 and an R^2 score of 0.754. XGBoost is also not interpretable as it aggregates shallow decision trees, but it achieved moderate performance. The linear ElasticNet and ridge regression models still outperform XGBoost for our dataset.

3.0.7 Neural Networks

We used Bayesian Optimization with Hyperopt to tune the hyperparameters of a neural network. The best configuration utilized ReLU with SGD optimizer and a batch size of 16. This resulted in an out-of-sample Mean Squared Error (MSE) of 4.076 and an R^2 score of 0.712. `Bayesian Optimization` enabled efficient exploration of the hyperparameter space and refinement of the model. The neural network model is also a black-box which is not interpretable, and it did not perform that well on our small dataset.

3.1 Regression Recommendation

The hyperparameter-tuned ElasticNet Regression model achieved the best performance, with an out-of-sample R^2 of approximately 0.7716. This is a marginal improvement over the baseline linear regression model, which achieved an out-of-sample R^2 of 0.7710. Ultimately, we recommend using ElasticNet as we retain 22 of the 27 features, so we can somewhat more easily identify the notable features relative to base linear regression. Furthermore, the coefficients for ElasticNet still have strong statistical properties and we can explain their effect on the outcome exam score value. ElasticNet maintains explainability with the best out of sample performance.

4 Classification Task

We tested 8 different models, logistic regression, support vector machines, k-nearest neighbors, naive bayes, decision trees, and ensemble methods such as random forest, XGBoost, and neural networks. We binned our final exam score into 4 ranges of low, average, good, and excellent with roughly even proportions. The model pros and cons as well as interpretability considerations are similar to what we discussed above with the linear regression models. The newly tested and best performed models of logistic regression and support vector classifiers will be discussed further. Table II presents the results of fine-tuned classification models, sorted by out-of-sample performance.

Table II: Fine-Tuned Model Performance Metrics

Model	**Accuracy (In-Sample)**	**Accuracy (OOS)**	**F1 (In-Sample)**	**F1 (OOS)**
Support Vector Machine	0.962	0.963	0.962	0.963
Logistic Regression	0.952	0.955	0.952	0.955
Neural Network	0.964	0.937	0.964	0.938
XGBoost	0.996	0.870	0.996	0.871
Random Forest	0.997	0.759	0.997	0.756
Naive Bayes	0.679	0.695	0.683	0.700
K-Nearest Neighbors	1.000	0.685	1.000	0.688
CART	0.855	0.655	0.854	0.655

We also highlight the baseline and tuned model Receiver Operating Characteristic Curves:

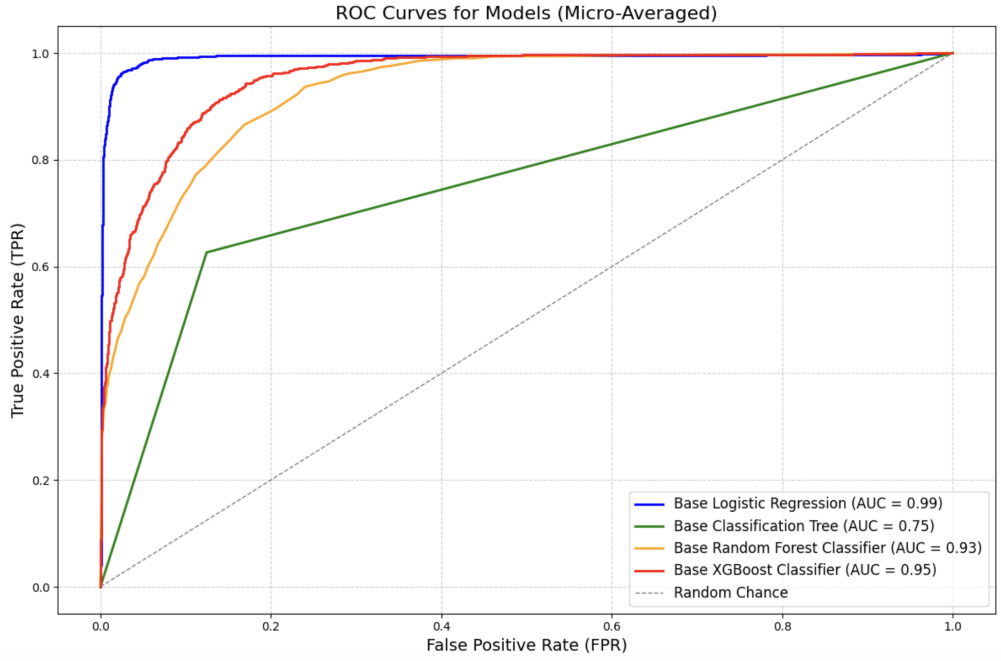

Figure 6: Baseline ROC Curves

The ROC curves reveal a clear improvement in model performance after fine-tuning. In the baseline models, Logistic Regression and Random Forest demonstrate strong discrimination capabilities with AUC scores of 0.93, while CART lags with an AUC of 0.75, highlighting its limitations in the baseline configuration.

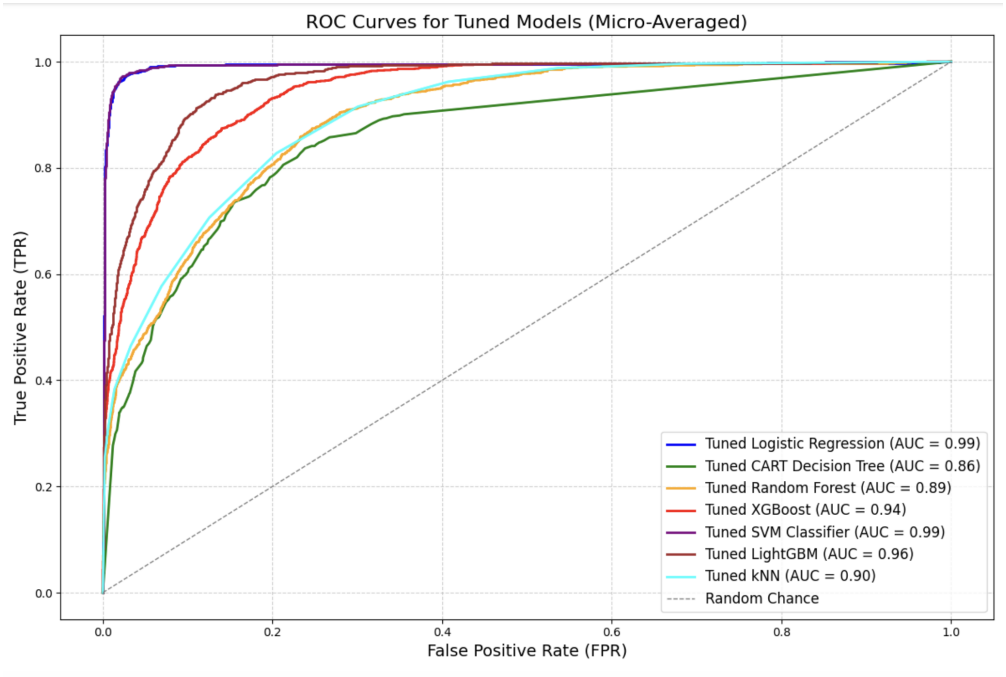

Figure 7: Tuned ROC Curves

After fine-tuning, nearly all models exhibit substantial performance gains, with Logistic Regression and SVM achieving near-optimal AUC scores of 0.99. The performance gaps between models are narrow, indicating that tuning mitigates variability in performance across algorithms. However, CART remains an under-performer even after tuning, suggesting inherent limitations in its suitability for this classification task. Similarly from table II, tuned models display improved consistency and effectiveness, with Logistic Regression, SVM, and Random Forest emerging as the most reliable classifiers.

4.1 Logistic Regression

Best parameters identified for Logistic Regression:

- **C**: 10
- **Penalty**: L1
- **Solver**: saga
- **Max Iterations**: 500

The logistic regression model appears to perform best initially without hyperparameter tuning relative to the other models. This model is reasonably interpretable as we can assess the direct impact of the features on the log odds of the outcome variable. We subsequently fine-tune the complexity parameter which essentially regularizes the coefficients, and we experiment with balancing the class weights. The fine-tuned model achieved an out-of-sample accuracy of 95.5% and an F1 score of 95.5%. This is a strong model choice for the schools to implement, as it balances some interpretability with high performance.

4.2 Support Vector Classifiers

Best parameters identified for SVM:

- **C**: 100
- **Kernel**: Linear
- **Gamma**: Scale

Support vector classifiers performed on par with the tuned logistic regression model, indicating both offer robust performance. It achieves an out-of-sample accuracy of 96.3% and an F1 score of 96.3%. SVCs can handle high-dimensional data well and create boundaries to maximize the margin between the support vectors and given classes. This model can also capture non-linear relationships depending on the kernel. SVCs are somewhat challenging to interpret as they are mathematically rigorous.

4.3 Classification Recommendation

We recommend Logistic Regression as the primary model for schools to predict student exam score ranges. This model achieves an out-of-sample accuracy of 95.5% and offers a strong balance between performance and interpretability. Its ability to provide probabilities for outcome categories allows for granular assessments of student performance, making it highly practical for identifying at-risk students. While SVM performed similarly, it lacks the same level of interpretability. Models like XGBoost and Random Forest seemed to overfit while CART underperformed significantly. Schools can use logistic regression predictions to guide interventions such as tutoring, counseling, or study groups, ensuring resources are effectively directed to improve outcomes and foster equitable learning opportunities.

5 Future Direction

With our recommendations, schools can the utilize ElasticNet regression model to predict exam scores and the Logistic Regression model to classify the performance range. These methods allow schools to determine the primary features affecting academic performance while maintaining strong performance potential. Consequently, they can identify at-risk students and provide additional support to improve academic outcomes. In the future, we would acquire more student data, further fine-tune the model parameters, and enhance machine learning with an optimization task to allocate resources to particular students. Our models provide a strong foundation to help school districts decide the best approach for their students to succeed.

6 Appendix

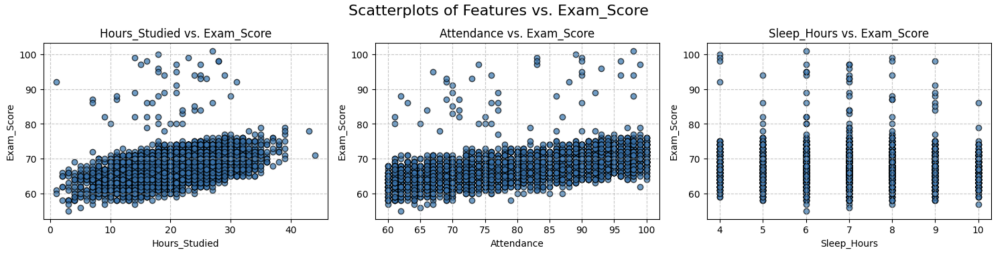

Figure 8: Relationships with Outcome

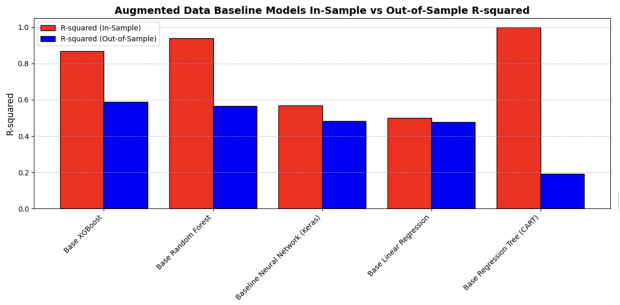

Figure 9: Augmented Data Baseline Results

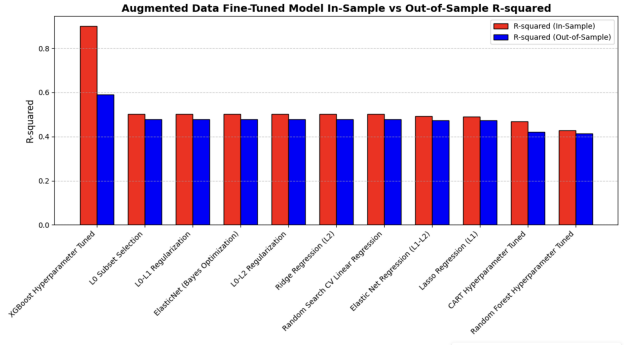

Figure 10: Augmented Data Tuned Results

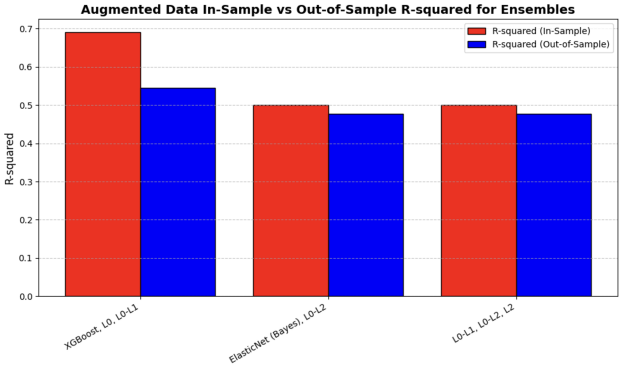

Figure 11: Augmented Data Ensemble Results

Model	R-squared (In-Sample)	R-squared (Out-of-Sample)	MSE (In-Sample)	MSE (Out-of-Sample)	MAE (In-Sample)	MAE (Out-of-Sample)
Base Linear Regression	0.7171461892305818	0.7710816276470018	4.350563974868887	3.2357777836235955	0.4968145083316668 5	0.4481122945667827
Baseline Neural Network (Keras)	0.7419821023941 04	0.7097357511520386	3.9685635067748 53	4.102906119173954	0.8519619121569558	0.8733091339942367
Base XGBoost	0.9296886920928955	0.7086007595062256	1.0814553429931282	4.11894913148925	0.4391705169244656	0.7834047239595029
Base Random Forest	0.9481766664280157	0.6856294752109406	0.7970927719962153	4.4436501512859 29	0.4132052980132 4496	1.0087972768532525
Base Regression Tree (CART)	1.0	0.1136948239206854 2	0.0	12.527987897125568	0.0	1.6913767019667172

Figure 12: Baseline Regression Table Results

Model	R-squared (In-Sample)	R-squared (Out-of-Sample)	MSE (In-Sample)	MSE (Out-of-Sample)	MAE (In-Sample)	MAE (Out-of-Sample)
ElasticNet (Bayes Optimization)	0.7169929179395296	0.7713632364664269	4.3529214349130685	3.231797222834672	0.5011012612954711	0.4511997732382 36
Ridge Regression (L2)	0.7171454297301 25	0.7711021860233069	4.3505756567172496	3.2354871894845942	0.4970726358320028 4	0.4483301681721621 5
Random Search CV Linear Regression	0.7171461892305818	0.7710816276470018	4.350563974868887	3.2357777836235955	0.4968145083316668 5	0.4481122945667827
Elastic Net Regression (L1-L2)	0.7046374024441397	0.7622645286767766	4.542961160589586	3.3604081165698347	0.6484821021459 92	0.5821731402907829
XGBoost Hyperparameter Tuned	0.7739548683166504	0.7500857710838318	3.4767921941207756	3.5325557634713762	0.6280698381061247	0.6767354292876782
Lasso Regression (L1)	0.6827688564764478	0.7436141810237071	4.879320455205 43	3.6240321322936 56	0.8317669539235267	0.7573543344957 84
Linear Recursive Feature Elimination	0.5966155123869918	0.6540507471903234	6.204441845970849	4.890017760464271	1.3021861502918586	1.2021216882188883
CART Hyperparameter Tuned	0.6478771403501996	0.6253663873016284	5.415988647114475	5.154122541603630 5	0.9473982970671713	1.2378971255673223
Random Forest Hyperparameter Tuned	0.5306666433649116	0.5711553077444302	7.218798950561008	6.061750804717776	1.5891870167607653	1.5388489479274958

Figure 13: Tuned Regression Table Results

Model	Accuracy (In-Sample)	Accuracy (Out-of-Sample)	ROC AUC (In-Sample)	ROC AUC (Out-of-Sample)	Precision (In-Sample)	Precision (Out-of-Sample)	Recall (In-Sample)	Recall (Out-of-Sample)
Base Logistic Regression	0.9468306527909177	0.9447806354009077	0.9927218298018233	0.9905352302809123	0.9470914480913717	0.9452963536133738	0.9468306527909177	0.9447806354009077
Base Neural Network	0.9379375591296121	0.9114977307110439	0.8947215434577439	0.894102811940291 3	0.9378844010036752	0.9124346511097908	0.9379375591296121	0.9114977307110439
Base XGBoost Classifier	1.0	0.7738275340393344	1.0	0.9481036812918376	1.0	0.7757976747439391	1.0	0.7738275340393344
Base Random Forest Classifier	1.0	0.7329803328290468	1.0	0.9196884733459462	1.0	0.7314736165636243	1.0	0.7329803328290468
Base Classification Tree	1.0	0.6263237518910741	1.0	0.7524368629418073	1.0	0.6289689042039778	1.0	0.6263237518910741

Figure 14: Baseline Classification Table Results

Model	Accuracy (In-Sample)	Accuracy (Out-of-Sample)	ROC AUC (In-Sample)	ROC AUC (Out-of-Sample)	Precision (In-Sample)	Precision (Out-of-Sample)	Recall (In-Sample)	Recall (Out-of-Sample)
Tuned SVM Classifier	0.9578051087984862	0.9546142208774584	0.9937986552976938	0.9917291356305191	0.9583437396638651	0.9551959845312727	0.9578051087984862	0.9546142208774584
Tuned Logistic Regression	0.9538315988647115	0.9531013615733737	0.9934750417285698	0.991483324344121	0.9540528115938838	0.9534828158333651	0.9538315988647115	0.9531013615733737
Tuned LightGBM	0.9996215704824977	0.8071104387291982	0.9999979234693857	0.9572947294906731	0.9996220138679899	0.8117178741847421	0.9996215704824977	0.8071104387291982
Tuned XGBoost	0.9973509933774835	0.7700453857791225	0.9999716712670625	0.9371976112417354	0.9973560617783077	0.7724179916129876	0.9973509933774835	0.7700453857791225
Tuned kNN	0.7623462630085147	0.6724659606656581	0.9348022332044392	0.8851457910149899	0.7592381362346076	0.6677642477850673	0.7623462630085147	0.6724659606656581
Tuned Random Forest	0.7165562913907285	0.6686838124054463	0.9153038348526291	0.8880151346563476	0.7060003832550236	0.6548705711495229	0.7165562913907285	0.6686838124054463
Tuned CART Decision Tree	0.7952696310312204	0.6437216338880484	0.957986126999123	0.8519324737936453	0.7951934549263677	0.6439071162255183	0.7952696310312204	0.6437216338880484

Figure 15: Tuned Classification Table Results