CS M148 Final Report

Sonal Aggarwal, Katherine Callahan, Genevieve Gonzales,
Connor Wang, and Masahiro Yamaguchi

December 13, 2024

1 Introduction

The data set used in this study is Student Performance Factors, with each observational unit representing an individual student. It contains 6,607 rows and 20 columns, including 7 numerical and 13 categorical variables describing the various factors influencing student performance. The target variable is exam score, a continuous measure of academic performance ranging from 1 to 100 (occasionally 101, possibly due to extra credit). The primary objective of this project is to build a model that predicts students' exam scores. The Student Performance Factors data set is a synthetic data set created to simulate real-world scenarios for analyzing factors influencing student academic performance. Below is an overview of the variables included in the data set and what they measure:

Hours_Studied	Number of hours spent studying per week.
Attendance	Percentage of classes attended.
Parental_Involvement	Level of parental involvement in the student's education (Low, Medium, High).
Access_to_Resources	Availability of educational resources (Low, Medium, High).
Extracurricular_Activities	Participation in extracurricular activities (Yes, No).
Sleep_Hours	Average number of hours of sleep per night.
Previous_Scores	Scores from previous exams.
Motivation_Level	Student's level of motivation (Low, Medium, High).
Internet_Access	Availability of internet access (Yes, No).
Tutoring_Sessions	Number of tutoring sessions attended per month.
Family_Income	Family income level (Low, Medium, High).
Teacher_Quality	Quality of the teachers (Low, Medium, High).
School_Type	Type of school attended (Public, Private).
Peer_Influence	Influence of peers on academic performance (Positive, Neutral, Negative).
Physical_Activity	Average number of hours of physical activity per week.

Learning_Disabilities	Presence of learning disabilities (Yes, No).
Parental_Education_Level	Highest education level of parents (High School, College, Postgraduate).
Distance_from_Home	Distance from home to school (Near, Moderate, Far).
Gender	Gender of the student (Male, Female).
Exam_Score	Final exam score.

The Student Performance Factors data set is highly relevant to our project. It provides a diverse range of variables that are sufficient to analyze and predict students' exam scores (Exam_Score). These variables represent a wide range of factors such as demographic, behavioral, social influences, and accessibility, all of which are likely to influence a student's academic performance.

The overarching goal of our project was to create a model to predict students' exam scores and gain insight into which features are the most important for this prediction. Thus, we focused on supervised learning, as our task was a prediction (specifically regression) task. As part of the project, the following questions were considered to guide the analysis, along with key assumptions to ensure the reliability and validity of our findings:

Questions for Prediction Task

- **Are all the variables equally important in predicting our target variable, Exam_Score?**
    - Most likely not. Logically, we expect variables like Hours_Studied and Previous_Scores to have a stronger correlation with Exam_Score, while variables like Gender and Distance_from_Home are likely to have less influence. To address this, we will use feature selection methods to identify the most influential predictors.

- **The data set is synthetic, does this impact the realism of the relationship between the variables?**
    - Although the data set is synthetic, it is designed to simulate realistic relationships between variables, enabling us to model more effectively. However, we will need to exercise caution in making inferences about the real world based on our model's predictions. Even if the model performs well on the validation and test sets in the split of the original dataset, it may not generalize to real data because it was not trained on real data.

- **Do missing values impact the quality of the dataset?**

    - The proportion of missing values is minimal, around 1% of the data. Removing rows with missing values is unlikely to significantly impact the model's performance, so this approach is acceptable.

- **Are the categorical variables encoded?**
    - The categorical variables are not numerically encoded in the original dataset, but we will encode them appropriately before building the model to ensure compatibility with our predictive methods.

<u>Questions about the Data</u>

- What is the source of this data? Was it collected by a government agency or a school district?
- Which subject is this for? Depending on the subject, the importance of various predictors may vary significantly.
- Which geographic region is the school located? Certain features may be more or less important depending on the geographic region.
- Which level of education is this for? Is it for elementary school, middle school, high school, or college? Given that Attendance is one of the variables, it may be for college. If so, were the lectures for this hypothetical class recorded, or was it a fully in-person class? If the lectures were recorded, then attendance might not tell the full story. While we do not have the answer to this question, attendance turned out to be one of the top predictors in our model, which may suggest that this data is for a hypothetical in-person class.
- How are "parental involvement," "motivation level," "peer influence," and "teacher quality" measured? Was it through a survey?
- While this data is synthetic, those are some questions we would ask about a real-world dataset with these features.

<u>Assumptions</u>

- The relationships in the synthetic data set mimic real-world relationships.
- The relationships between predictors and the target variable follow patterns observed in real-world student performance.
- Variables are measured consistently.
- The data set is assumed to be free from bias that could disproportionately affect certain variables.
- The missing values are assumed to occur randomly and do not affect certain groups or observations.

2 Methodology

Data Cleaning and Preprocessing

The Kaggle API was used to load the data as a CSV file. Upon loading the data, we observed the first 20 entries as well as a random sample of 20 other entries and did not identify any issues.

Upon getting a statistical description of each column, we did not see clear evidence of any inconsistent or invalid values. The only anomaly we identified was a maximum exam score of 101. Given that this is a synthetic dataset, there is no way to verify the true range of exam scores, but we assume that this is due to extra credit, and thus we did not exclude the data point from the dataset.

We identified that a total of 235 values were missing from the dataset, particularly from the Teacher Quality, Parental Education Level, and Distance from Home columns with respective missingness proportions 0.011806, 0.013622, and 0.010141. Due to the low proportion of missingness, not a significant amount of data would be lost by simply dropping rows with missing values, so that was the strategy we used. No duplicates were identified. Since all students were included in the original data and there were no duplicates, we concluded that the data is complete.

Upon inspecting the columns, we also noticed that the column names were clear and intuitive, so we decided not to rename them. The data types were consistent for each column, enabling us to determine that the dataset is tabular and tidy. Also, the data was not nested, so we did not need to join or merge data.

The main preprocessing step needed was encoding the categorical variables. Since the dataset contained both ordinal and binary variables, we used ordinal and label encodings. For categorical variables with an inherent order, Ordinal Encoding was used to assign numerical values based on the specified order. For Parental Involvement, Access to Resources, Motivation Level, Family Income, and Teacher Quality, “Low” was encoded as 0, “Medium” was encoded as 1, and “High” was encoded as 2. For Distance from Home, “Near” was encoded as 0, “Moderate” was encoded as 1, and “Far” was encoded as 2. For Peer Influence, “Negative” was encoded as 0, “Neutral” was encoded as 1, and “Positive” was encoded as 2. For Parental Education Level, “High School” was encoded as 0, “College” was encoded as 1, and “Postgraduate” was encoded as 2. For categorical variables with only two categories, Binary Encoding was applied using LabelEncoder. For Extracurricular Activities, Internet Access, and Learning Disabilities, “No” was encoded as 0 and “Yes” was encoded as 1. For Gender, “Male” was encoded as 0 and “Female” was encoded as 1. For School Type, “Public” was encoded as 0 and “Private” was encoded as 1. These encoding strategies enabled us to retain the ordinal and binary relationships in the original categorical features while ensuring compatibility with the algorithms we used.

Another preprocessing step was standardizing the data. For almost all of the methods we tried except for decision trees, we standardized the data. This is because the features of the dataset were in quite different scales, especially due to the mix between categorical and continuous variables. Standardization was not strictly necessary for linear regression, as when we ran the Least Squares algorithm with both standardized and non-standardized data, the results were virtually identical. However, we still standardized the data for linear regression in order to be able to try ridge and lasso regression, which require standardization so that features with large scales do not have their coefficients unfairly penalized (e.g. "Attendance" is on a 0-100 scale whereas "Extracurricular Activities" is on a 0-1 scale, so standardizing ensured that the regularization penalty would not over-penalize Attendance compared to Extracurricular Activities.) Even more importantly, standardization enabled us to compare the coefficients in order to determine feature importance for linear regression, which was a main goal of our project. While linear regression only required the predictors to be standardized, neural networks required the response variable to be standardized as well in order to ensure proper computation of gradients for gradient descent. When standardizing the data, we were careful to standardize after the data had been split in order to prevent data leakage.

Overall, the dataset was already mostly clean, so the only cleaning and preprocessing steps required were removing missing values, encoding categorical variables, and standardizing the data.

Exploratory Data Analysis

To conduct Exploratory Data Analysis, we initially started by exploring the relationship between hours of sleep and exam score. First we created a histogram to examine the distribution of hours of sleep, and we observed that this feature seemed to be normally distributed with a mean of 7. Since the number of hours of sleep were discrete while the exam score was continuous, we created a boxplot to investigate the relationship between the two variables. The boxplot revealed several outliers in the data, but excluding outliers, the spread of exam scores was quite similar for each hour. There did not seem to be a discernible relationship between Sleep Hours and Exam Score, highlighting the need for robust feature selection techniques.

An important part of Exploratory Data Analysis was conducting correlation screening. We created a correlation matrix of all the variables in order to identify which variables had the highest correlation with the response variable (Exam Score) as well as to identify possible collinearity among predictors. Most pairs of variables were negligibly correlated, with correlations ranging from around 0.01-0.02. Pairs of predictors did not have any non-negligible correlations, so we concluded that there was no evidence of collinearity. The only non-negligible correlations observed were between Exam Score and the predictors Attendance (0.58), Hours Studied (0.45), Access to Resources (0.17), Previous Scores (0.17), Parental Involvement (0.16), Tutoring Sessions (0.16), Parental Education Level (0.11), Peer Influence (0.10), Motivation Level (0.09), Family Income (0.09), Distance from Home (-0.09), Teacher Quality (0.08), Learning Disabilities (-0.08), and Internet Access (0.05). Since most of these correlations were also quite low, correlation screening suggested keeping only Attendance and Hours Studied as predictors. This was our initial strategy, but it turned out to be ineffective, as discussed in the next section.

Several methods we tried, including linear regression and neural networks, involved splitting the data into a training, validation, and test set. We decided to split the data into these three distinct sets as opposed to only training and test sets because several of our techniques, such as regularization and cross-validation, involved fitting the validation data, so we wanted to ensure that we did not overfit to the validation set and had a separate set for an unbiased evaluation of the final model. We used a 70/20/10 split, which is a common split used for regression tasks. We could have also used a 60/20/20 split, but we decided to use 70/20/10 in order to have more training data. We used the train_test_split function from sci-kit learn to conduct the split, which randomly shuffles the data. Shuffling helps in creating a representative sample of the data in each split, preventing bias. We did not use stratified sampling since our response variable is continuous, and stratification is more important for classification tasks. The test set was kept aside and was not used during model training or hyperparameter tuning in order to retain it as a way to evaluate the final model on completely unseen data. Although standardization was a preprocessing step, we included it as part of data splitting in order to prevent data leakage. Specifically, we did not want to encode information (i.e. mean and standard deviation) about the validation or test data in the training data, as such leakage of information could lead to overfitting.

Main Regression Task

We experimented with several different methods for modeling the data, including linear regression, logistic regression, decision trees, PCA, and neural networks. Of these, linear regression and neural networks performed the best for our task, and they performed about equally well as one another, achieving an R^2 score of roughly 0.7. The neural network is described further in the appendix. We ultimately decided on linear regression as our main model because it is far more efficient compared to the neural network and enables us to more easily compare feature importance.

We conducted forward selection in order to identify a set of predictors to use. The output of forward selection were the 9 features Hours Studied, Attendance, Parental Involvement, Access to Resources, Previous Scores, Motivation Level, Tutoring Sessions, Family Income, and Parental Education Level. The training R^2 score was 0.74 and the MSE was 3.84. When we observed these results, we were initially hesitant to use all nine of these features because a high number of features can usually lead to overfitting. Thus, we initially conducted linear regression with only the two features Hours Studied and Attendance, which yielded a R^2 score of 0.34 on the training set and 0.30 on the validation set. This was significantly worse than the metrics with

9 features, so we tried including the two features with the next highest correlation with Exam Score, which were Previous Scores and Tutoring Sessions, resulting in an R^2 score of 0.55 on the training set and 0.50 on the validation set, a considerable improvement from the model with just two features. However, this was still worse than the performance of the model with all 9 features. We decided to collect more metrics about the performance of the model with all 9 features and observed the following:

Metric	Training Performance	Validation Performance
rMSE	1.96	2.39
MAE	0.46	0.54
MAD	0.27	0.29
Corr	0.86	0.81
R^2	0.74	0.65

While the validation performance is worse than the training performance, it is not substantially worse to the point of overfitting and is still better than the validation performance with the models using only 2 or 4 features. According to the validation metrics, using all 9 features appears to explain about 65% of the variance in Exam Scores while using 2 or 4 features only explains 30% and 55%, respectively. Therefore, we decided to use all 9 features.

While these metrics did not indicate obvious overfitting, we still wanted to try regularization techniques to see if they might improve performance on the validation set by penalizing large coefficients. We ensured the data was standardized before conducting ridge and lasso regularization so that coefficients corresponding to variables with larger scales (e.g. Attendance) would not be unfairly penalized. To determine the regularization strength λ, we conducted hyperparameter tuning using k-fold cross-validation. We explored a range of 100 λ values from 10^-1 to 10^6. For each value of λ, we conducted 10-fold cross-validation using the training set with R^2 as the scoring metric. The average R^2 value among the 10 folds was recorded for each value of λ. We selected the λ value corresponding to the maximum average R^2 value as the regularization strength to fit the final ridge regression model with. Finally, we obtained the validation set R^2, coefficients, and intercept for ridge regression to compare the model with normal linear regression. We observed that the R^2 score, coefficients, and intercept were virtually identical to several decimal places. The negligible impact of ridge regularization corroborates our earlier judgement about overfitting not being a significant concern for our model – the normal linear regression model is already able to generalize fairly well to the validation set. It also supports our earlier finding through correlation screening about the lack of multicollinearity of the data, as if there was multicollinearity among variables, we would expect ridge regression to improve the results.

In addition to ridge regression, we also experimented with lasso regression. Since lasso regression can drive coefficients to zero, it can be used as a feature selection technique and can be used to discern which features may not be contributing as meaningfully to the response variable. Our objective was to see if lasso regression would indicate if some of the 9 selected features were unnecessary and possibly leading to overfitting in order to test our earlier assumption that including all 9 features was the best choice. We used the same hyperparameter tuning and cross-validation procedure to determine the regularization strength. The results for lasso regularization were slightly different compared to normal linear regression:

	Lasso Regression	Normal Linear Regression
R^2	0.62	0.63
Coefficients	[1.65, 2.20, 0.58, 0.59, 0.57, 0.29, 0.53, 0.32, 0.27]	[1.75, 2.31, 0.695, 0.69, 0.68, 0.39, 0.63, 0.42, 0.38]
Intercept	67.24	67.24

The R^2 score was very slightly worse for lasso regression compared to normal linear regression, and the coefficients had a difference of about 0.1 among the two algorithms. Since none of the coefficients were driven to zero, lasso regression did not screen out any of the 9 features, and similar to ridge regression did not improve the performance on the validation set. This further supports our decision to use all 9 features and further indicates that overfitting is not a concern. Thus, we decided to proceed with normal linear regression for the regression task as opposed to ridge or lasso regression.

3. Results

Model Evaluation

We evaluated our model using the PCS framework. As linear regression was quite efficient for the task at hand, we are already certain about Computability. Thus, we investigated Predictability and Stability.

In order to obtain more validation sets to evaluate the model, we conducted k-fold cross-validation with 10 folds. For each fold, we computed the rMSE, MAE, and R^2 score.

Finally, we reported the average of these metrics across all 10 folds. The results compared to the output of linear regression on the original validation set are as follows:

	Original Validation Set Performance	Cross-Validation Performance
rMSE	2.48	2.12
MAE	0.92	0.84
R^2	0.63	0.72

The metrics for cross-validation are better than those for the original validation set. This suggests that the original validation set might contain a particularly challenging subset of data, despite the random sampling used in the train-test split. The cross-validation metrics are generally more reliable since they are averaged across multiple validation sets, providing a more comprehensive view of the model's performance. Thus, the results of cross-validation suggest that the model is robust and generalizes well across the dataset, explaining on average 72% of the variance in exam scores. The slightly worse metrics on the original validation set highlight the importance of using cross-validation as a technique to evaluate model performance. Since cross-validation provided enough insight about the model's performance, we did not conduct leave-one-out cross-validation as it would be highly inefficient.

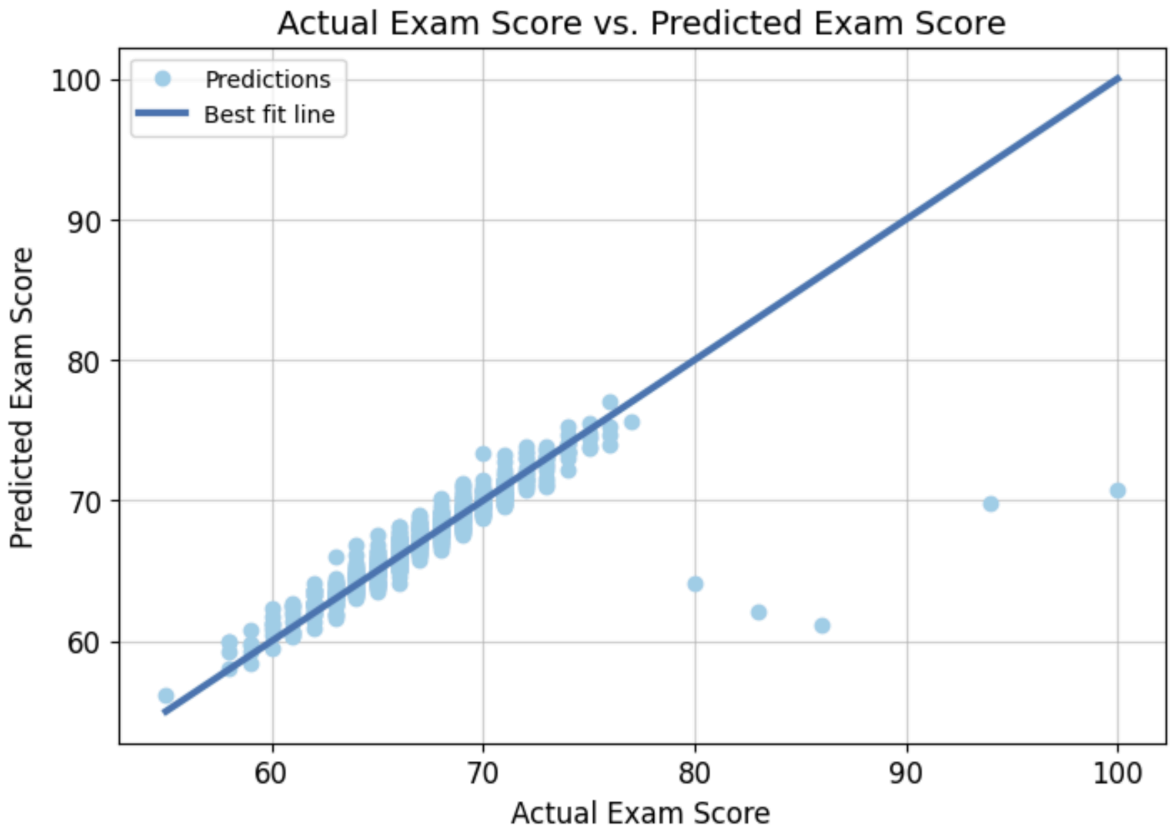
Next we conducted bias-variance decomposition in order to gain insight into how much of the error can be attributed to bias vs. variance. The square root of the overall MSE was 2.485, the average bias was 2.483, and the average standard deviation was 0.103. This decomposition indicates that the vast majority of the error is due to bias rather than variance. The low variance indicates that the model's predictions are stable across different datasets. Thus further corroborates how overfitting is not a concern for our model despite the use of 9 predictors.

To assess the final performance of the model on the unseen test set, we began by retrieving the rMSE, MAE, MAD, correlation, and R^2 metrics.

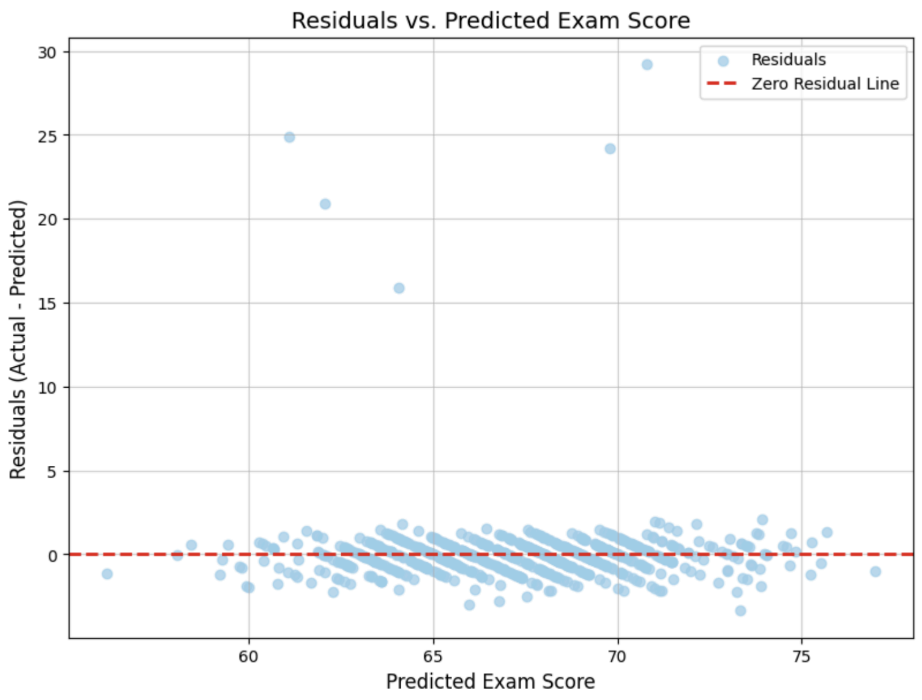
	Validation	Test
rMSE	2.39	2.25
MAE	0.54	0.87
MAD	0.29	0.58
Correlation	0.81	0.83
R^2	0.65	0.69

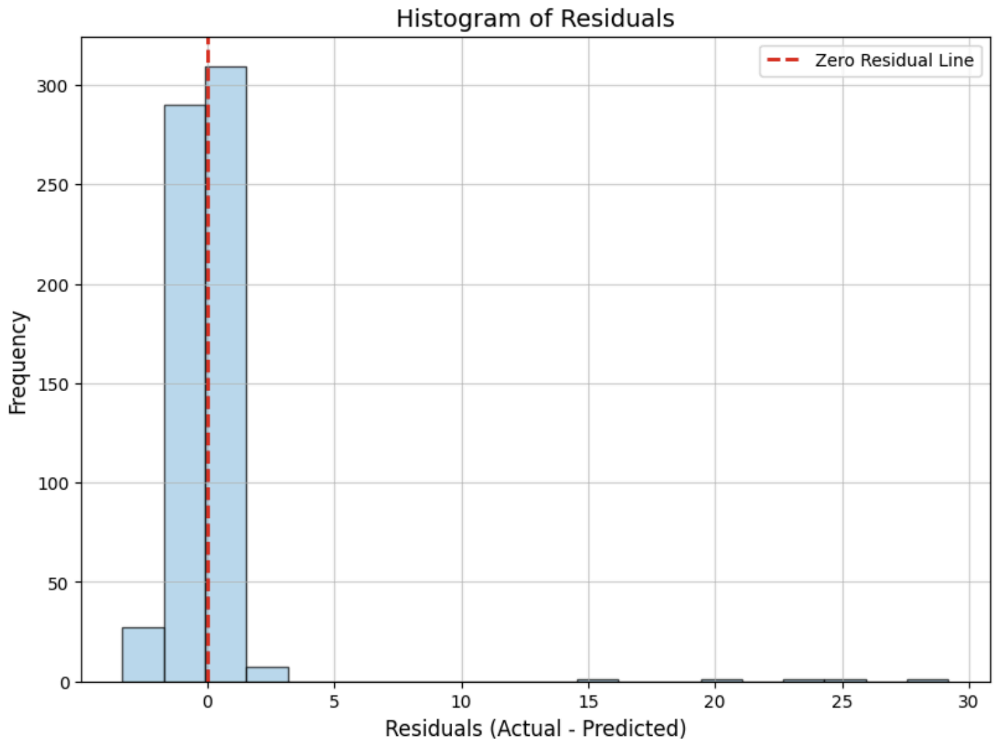
Overall, the metrics are quite similar on the test set compared to the validation set, with a slight improvement in rMSE, correlation, and R^2 and a slight worsening in MAE and MAD. This indicates that the model's performance is stable across different datasets, meaning it is not overfitting.

Plotting the predicted exam score against the actual exam score gave significant insight about the model's performance.



We see from the plot that there is a tight linear relationship between the predicted exam score and the actual exam score aside from 5 points which appear to be outliers. To further investigate these outliers, we plotted the residuals against the predicted exam score and created a histogram of the residuals.


Residuals vs. Predicted Exam Score
Residuals
Zero Residual Line
Residuals (Actual - Predicted)
Predicted Exam Score
30
25
20
15
10
5
0
60
65
70
75


Histogram of Residuals
Zero Residual Line
Frequency
Residuals (Actual - Predicted)
300
250
200
150
100
50
0
0
5
10
15
20
25
30

The scatterplot of Predicted Exam Score vs. Residuals does not reveal a relationship between the residuals and exam scores. Thus, the model's predictions do not seem to be biased in a particular direction. There is no evidence of heteroscedasticity.

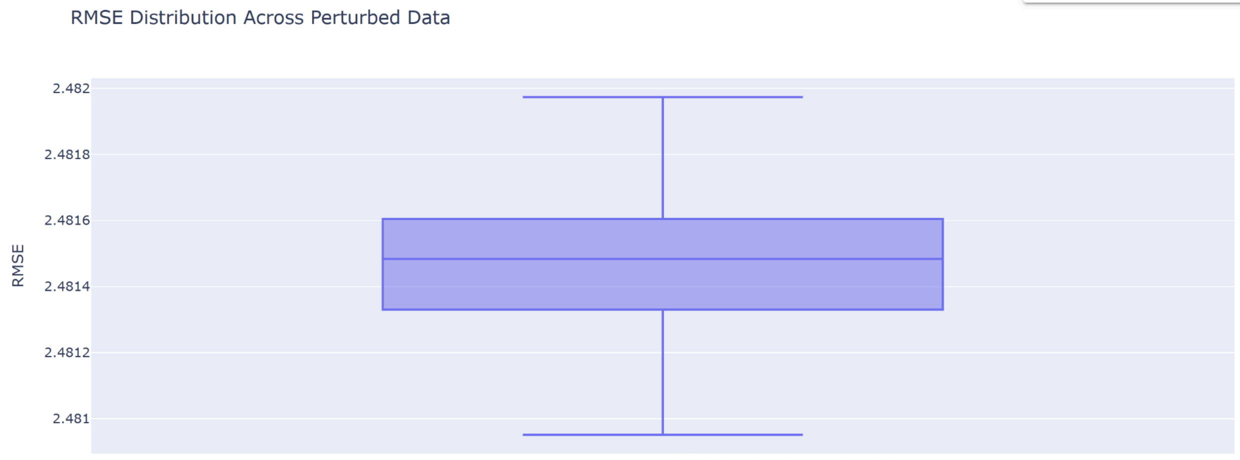
The histogram of the residuals reveals that the residuals are roughly normally distributed close to 0. There are only 5 residuals with an absolute value greater than 5, and they appear to correspond to the 5 "outlier" points from the first scatterplot. If those 5 points are not considered, the remaining residuals are quite small, with an absolute value less than 5. Thus, the next step was to obtain the metrics for the model when the outliers, defined as data points in which the residual has an absolute value greater than 5, are excluded. We thought it was reasonable to observe the model's metrics excluding these points because they comprise less than 1% of the test data.

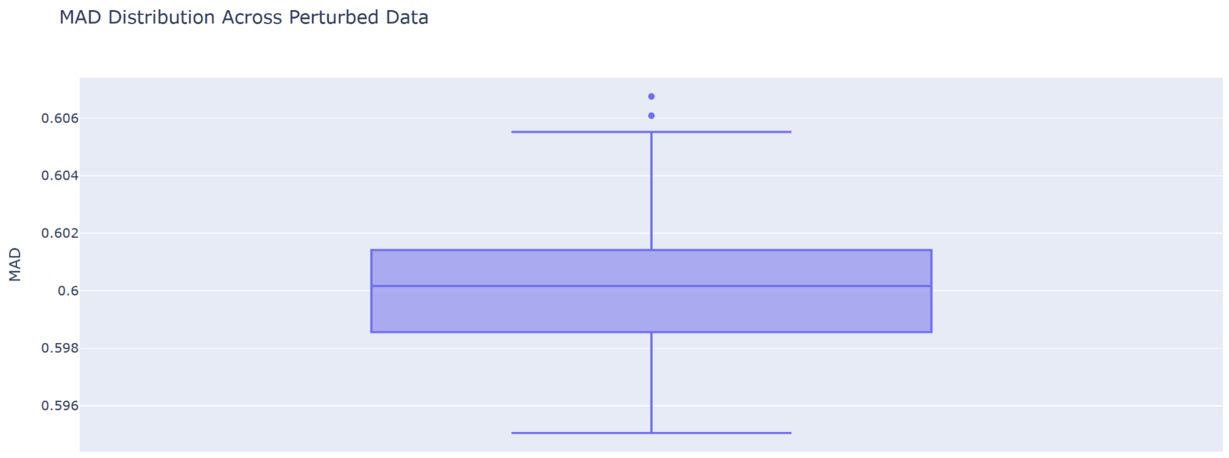
Metrics	Test	Test Without Outliers
rMSE	2.25	0.87
MAE	0.87	0.69
MAD	0.58	0.57
Correlation	0.83	0.97
R^2	0.69	0.94

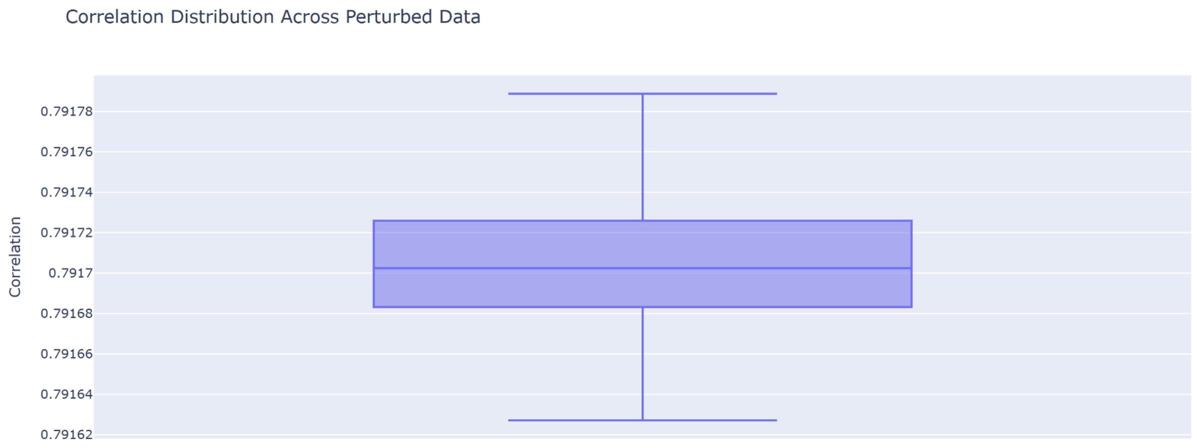
The results indicate a substantial improvement in metrics on the test set without the 5 outliers. Naturally, the improvement is not as great for MAE and MAD since those metrics are more robust to outliers because they do not square the error. The improvement in R^2 is especially promising, as once outliers are excluded, the model is able to explain 94% of the variance in exam scores. The tight correlation and high R^2 indicate the model performs well for the vast majority of data points (>99%).

Thus, it appears that most of the bias stems from outlier points. The significant difference between the metrics with and without outliers suggests that the outliers have a disproportionate effect, obscuring the model's overall performance. The residuals of the outlier points are all positive, so it appears that the model predicts these students would score much lower than they actually do. These outliers may reflect students who perform better than expected due to unmodeled factors. This is discussed further in the Conclusion section.

Finally, to evaluate the Stability of the model, we introduced some noise into the data and observed how the model's predictions fluctuated with the added noise. One of the main predictors in the model was Attendance, which can often have noise in the real world, as methods to take attendance may not always be accurate. To stress-test the model's results, we randomly sampled 30% of the rows to have added noise up to 50% of the standard deviation in Attendance. We took 100 such bootstrapped samples to simulate further variation in the data. We retrained the model on each of the 100 perturbed bootstrapped datasets and evaluated its performance on the original validation set. We visualized the distribution of each metric across the 100 datasets using boxplots in order to assess the variability in the model's performance under perturbed conditions.







Considering the very narrow ranges on the y-axes, the metrics are barely fluctuating among the 100 perturbed bootstrapped datasets.

Metrics	Original Validation Set	Median Among Perturbed Samples
rMSE	2.39	2.48
MAE	0.54	0.93
MAD	0.29	0.60
Correlation	0.81	0.79

The metrics are fairly similar to those of the original validation set. The fact that the median metrics among the 100 perturbed bootstrapped samples are not significantly worse than the original validation metrics, as well as the fact that the metrics hardly fluctuate across the 100 samples, indicate that the model is stable in the face of perturbations and noise. Furthermore, the consistency of the metrics also implies that the model is not overfitting. Rather than memorizing specific data points, it is capturing the true underlying patterns in the data.

Model Interpretation

One of our main goals in creating a regression model was to determine which features are most important in predicting exam scores. Since we standardized the data, we can directly compare the coefficients by their absolute value.

```
Feature Importance
                    Feature  Coefficient  Absolute Importance
0                Attendance     2.306846             2.306846
1             Hours_Studied     1.748431             1.748431
2      Parental_Involvement     0.695244             0.695244
3       Access_to_Resources     0.693329             0.693329
4           Previous_Scores     0.677704             0.677704
5         Tutoring_Sessions     0.627687             0.627687
6             Family_Income     0.419116             0.419116
7          Motivation_Level     0.387613             0.387613
8  Parental_Education_Level     0.375404             0.375404
```

As we observed from the correlation matrix earlier, Attendance and Hours_Studied are the most important predictors in predicting the Exam Score. Keeping all other predictors constant, for every standard deviation increase in the percentage of classes attended, a student's predicted exam score increases by about 2.3%. Keeping all other predictors constant, for every standard deviation increase in the number of hours studied, a student's predicted exam score increases by about 1.74%. All other predictors also have a positive association with the response variable – higher parental involvement, better access to resources, higher previous scores, more tutoring sessions, higher family income, higher motivation level, and higher parental education level are all associated with a higher exam score, keeping other features constant. However, the coefficients for those remaining variables are lower, indicating that they are not as important in predicting exam scores compared to Attendance and Hours Studied.

The y-intercept is about 67.24, which is the predicted Exam Score when all the predictors in standard units are 0. Since all the predictors are standardized, the 0-value is the mean, so the y-intercept is the score a student who has average values for every predictor would be predicted to receive on the exam.

We can represent the response variable, Exam Score, as a linear combination of the predictors: Predicted_Exam_Score = 2.31 * Attendance + 1.75 * Hours_Studied + 0.70 * Parental_Involvement + 0.69 * Access_to_Resources + 0.68 * Previous_Scores + 0.63 * Tutoring_Sessions + 0.42 * Family_Income + 0.39 * Motivation_Level + 0.38 * Parental_Education_Level + 67.24

Conclusion

Overall, our linear regression model is quite robust with reasonably strong predictive power, as it is able to explain about 62-71% of the variance in Exam Scores depending on the validation data used, increasing to 94% once outliers comprising less than 1% of the data are excluded. Thus, due to its reasonably strong predictions, the model does not appear to be underfitting. As demonstrated by ridge regularization, lasso regularization, k-fold cross-validation, bias-variance decomposition, residual analysis, and stability in the face of perturbation, the model is not overfitting despite including 9 predictors. The fact that including all 9 predictors significantly improved the model compared to only including the two strongest predictors Attendance and Hours Studied suggests that student performance is a multifaceted issue which cannot be explained by just one or two predictors. The 9 main predictors of student performance span multiple domains, including effort-based factors, socioeconomic status, and social support – all of these play a role in student performance and all must be taken into account. That being said, effort-based predictors (namely, attendance and hours studied) seem to be more deterministic of student performance compared to socioeconomic factors such as family income and parental involvement. This paints an encouraging picture for student performance, as while students may be disadvantaged to factors outside of their control, they can compensate for such disadvantages through the factors which are in their control, while students who are socioeconomically advantaged must still put effort into their grades. Thus, schools and educators could prioritize interventions targeting effort-based behaviors, such as improving attendance or encouraging consistent study habits.

The multifaceted nature of student performance also means that while the identified features predict exam scores fairly well, students' outcomes are not set in stone. In particular, there are "outlier" students who still perform highly even if the odds seem to be against them according to the model. This can be due to unmeasured variables such as innate talent, effective teaching, familiarity with the subject, strong test-taking skills, or pure luck. Human behavior is complex, and it is unlikely that any model, even one that can model more complex relationships such as a neural network, would be able to predict student performance with near certainty because of these extra factors.

However, the interpretation and implications of these findings must be treated with caution because the dataset is synthetic, meaning that the real world applicability of the model may be limited. An important area for further research and improvement would be further investigating these outliers and exploring why the model predicts them to score much lower than they actually do. It might also be helpful to collect data on some of these external factors which could be leading to these outliers, such as cognitive ability or student mental health. To be able to apply findings to real-world policy and academic interventions, it would be necessary to collect real-world data as opposed to using a synthetic data set, ideally spanning different school subjects, school districts, modalities of teaching, and geographical areas.

In conclusion, based on analysis of a synthetic dataset, it appears that students have significant agency in their academic success, but systemic support can enhance their opportunity to succeed. Academic performance is based on various interconnected factors, some of which are under students' control and some which are not, underscoring the complexity of human behavior and the need for nuanced approaches to improving outcomes.

4 Using Code

The code is split into several functions, including load_data to load the dataset, clean_data to remove missing values, preprocess_data to encode categorical variables, split_data to split the data into training, validation, and testing sets, forward_selection to perform forward feature selection and output the selected features and associated performance metrics, train_linear_regression to actually train the linear regression model and output the validation metrics and coefficients, ridge_regression and lasso_regression to perform regularization and output the associated validation metrics and coefficients, k_fold_cross_validation to perform cross-validation and output the resulting metrics, bias_variance to perform bias-variance decomposition and output the results, evaluate_test_performance to output the test set metrics, evaluate_stability to output boxplots and metrics indicating the model's performance on perturbed bootstrapped samples of the original data, rank_features to rank the coefficients for determining feature importance, and final_pipeline which puts everything together. The Jupyter notebook is organized into clear sections and includes comments describing each step.

If a user simply wants to run linear regression on the data and observe the validation, cross-validation, and test set metrics as well as obtain the ranked coefficients, they only need to call the final_pipeline() function in the Final Pipeline section of the Jupyter notebook which will run a pipeline including data cleaning, preprocessing, data splitting, stepwise variable selection, linear regression, k-fold cross-validation, and test set evaluation. If the user would also like to see intermediate steps such as trying ridge regression, lasso regression, bias-variance decomposition, plotting residuals, and evaluating stability in the face of perturbations, they can step through the code cell-by-cell in the Running the Code section of the notebook, which individually calls each function and displays the output. It is not necessary to upload any files to the environment in order to run the code, as the code uses kagglehub to load the dataset from Kaggle in the load_data function that is part of the pipeline.

Appendix

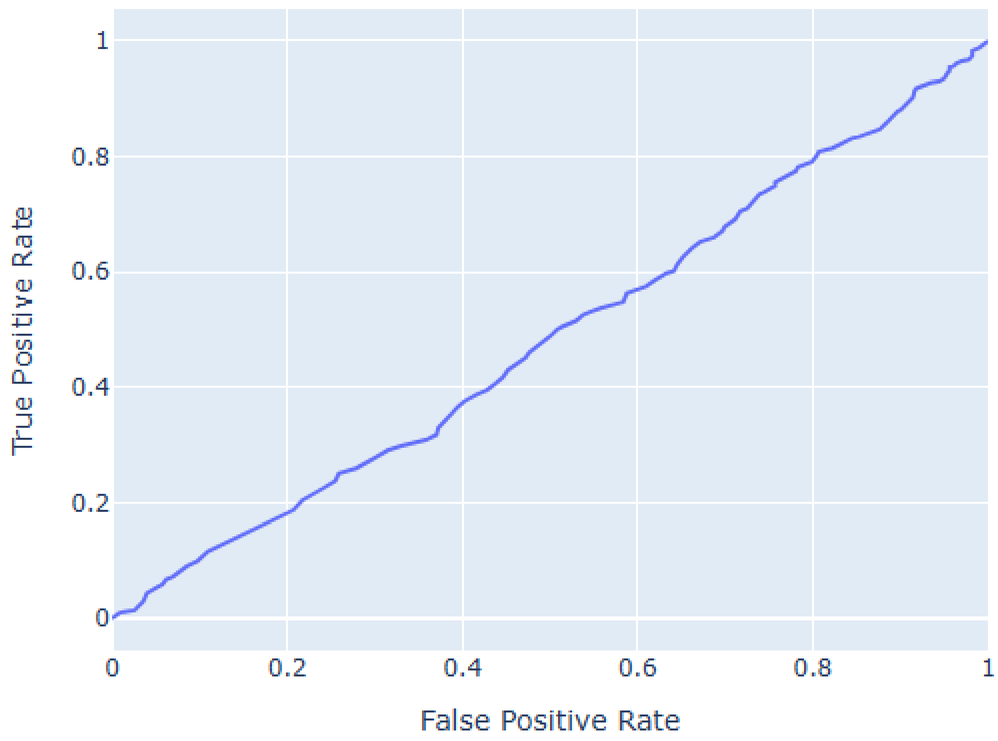
I. **Explain the exploratory data analysis that you conducted. What was done to visualize your data and split your data for training and testing?** This is explained in the main report.

II. **What data pre-processing and feature engineering (or data augmentation) did you complete on your project?** This is explained in the main report.

III. **How was regression analysis applied in your project? What did you learn about your data set from this analysis and were you able to use this analysis for feature importance? Was regularization needed?** Linear regression was the main method we used in our model, so this is explained in the main report.

IV. **How was logistic regression analysis applied in your project? What did you learn about your data set from this analysis and were you able to use this analysis for feature importance? Was regularization needed?**

Our logistic regression model used two predictor variables, family income and attendance, and a binary response variable, extracurricular activities (whether a student participates in extracurricular activities or not). Since family income was a categorical variable taking on one of three values, low, medium, or high, the values were converted to 0, 1, and 2 respectively in order to apply logistic regression. The attendance variable was kept as is. The resulting model produced a prediction accuracy of 49.76%, with a true positive rate of 50.52% and a true negative rate of 48.61% at a threshold of 0.5. The resulting ROC curve is shown below.



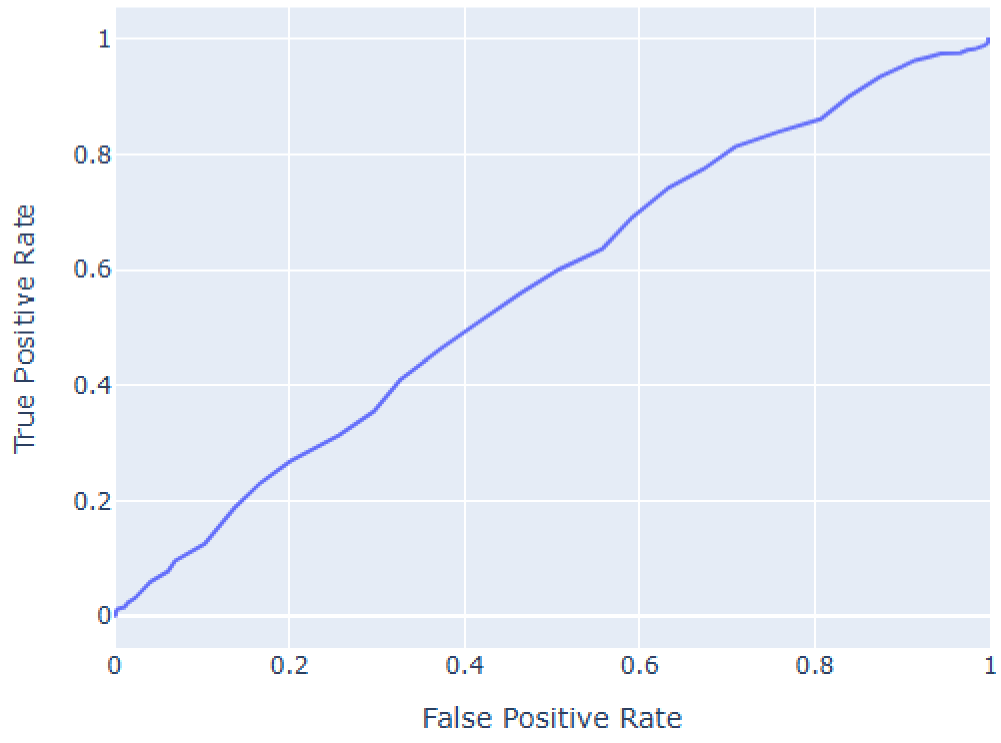
When choosing a threshold, we would ideally like a threshold which has a low false positive rate and high true positive rate. However, looking at the ROC curve, we saw that every single threshold tested produces a result where TPR essentially equals FPR. Therefore, all of the thresholds were equally as bad as each other. This was reflected in our cross validation, where the AUC and accuracy stayed around 50%, indicating our model was not making any meaningful predictions. Overall, we learned there was no meaningful linear relationship between attendance/family income and the probability of a student participating in extracurricular activities. Therefore, we were unable to use this analysis for feature importance. Regularization was not needed, as the problem lied in the fact that there was no linear relationship between variables, not that there was overfitting.

While we had initially tried using only Family Income as a predictor for Extracurricular Activities, that had resulted in an accuracy of about 63.4%, yet revealed a TPR of 100% and a TNR of 0%. This was a situation where just the accuracy did not tell the full story, as examining the TPR and TNR told us that the model was simply predicting "yes" for all of the samples since the majority of students participated in extracurricular activities. Thus we added the continuous variable attendance in hopes it would address the issue of the model always predicting "yes," but as we just discussed, the performance of the updated model was still very lacking.

The lack of effectiveness of logistic regression was expected for our task, as our main goal was to create a model to predict Exam Score, which is a continuous variable. In order to use logistic regression, we had to choose a categorical variable as a response variable, and we had seen earlier from the correlation matrix that there was no collinearity, meaning the predictors (the variables aside from Exam Score) were not correlated with one another.

V. **How were KNN, decision trees, or random forest used for classification on your data? What method worked best for your data and why was it good for the problem you were addressing?**

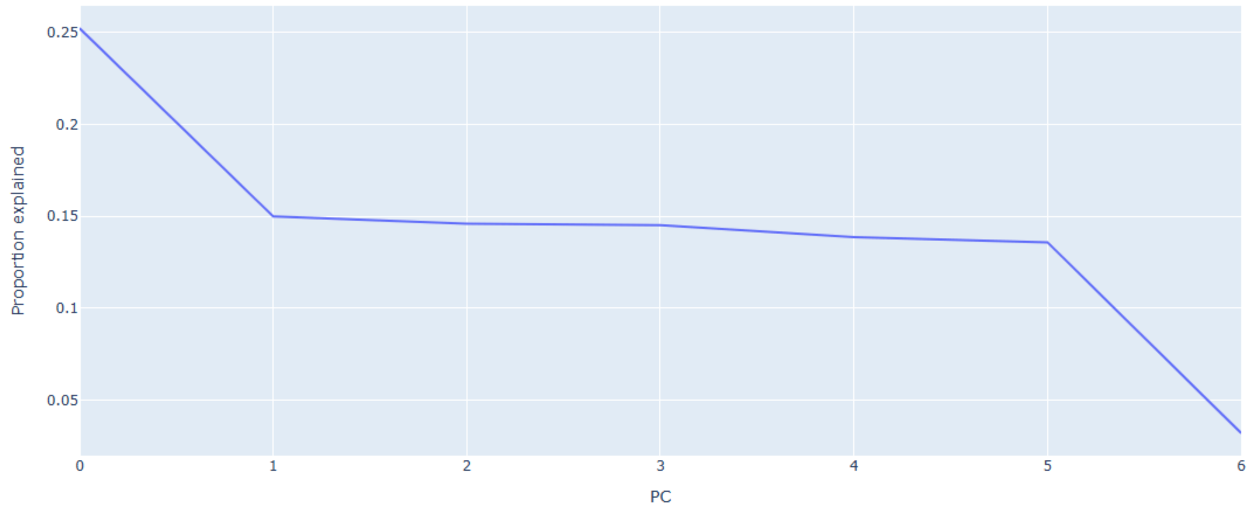
A random forest model was chosen to predict the binary response variable extracurricular activities based on a subset of the remaining features, chosen using the feature selection algorithm from the RandomForestClassifier from scikit learn. Before training the model, we dropped rows with missing values, converted categorical variables to numerical values, scaled and mean-centered values, and checked for collinearity amongst predictor variables (for which we found none). Then, a subset of features were selected (Hours studied, attendance, sleep hours, previous scores, tutoring sessions, physical activity, exam score) and the model was trained. The resulting ROC curve for our validation set is shown below.



The AUC is 0.567. While this model performed slightly better than our logistic regression model, the performance is still too low to be useful or draw any conclusions. When performing 5 fold cross validation on our validation set, we get similar results, with AUC taking on values between 0.49 to 0.53 and accuracies between 0.55 and 0.60. Even though performance was low, this method worked best for our data for several reasons. The problem we were addressing with this model was trying to choose a subset of all of our features in order to predict extracurricular activities. Due to the high number of features and nonlinear nature of our data, overfitting is a significant issue. The reason random forest was the best model for our problem was because it is an ensemble method, which creates multiple trees trained on multiple bootstrapped datasets, preventing the overall model from being overfit to a particular dataset. This is also the reason we chose this method over KNN and decision trees because of those model's risk of overfitting, especially with a noisy dataset such as ours.

VI. How were PCA and clustering applied on your data? What method worked best for your data and why was it good for the problem you were addressing?

We chose to apply PCA to our dataset. We began by filtering our dataset to only include numerical values, then standardizing the values such that the mean was 0 and standard deviation was 1 for each variable. We then applied SVD on the entire dataset. Analyzing the variability explained by each PC, we saw that the first principal component explained the most variability, with 25%, and the following 5 PCs each explained around 15%. Looking at the loading factors for PC 1, 2, and 3, we saw significant overlap in the variables with higher loading factors. For example, tutoring sessions and attendance had significant loading factors in all 3. This told us that PCA is not exactly "grouping" variables as we would've liked. Rather, every variable was contributing to the PCs with varying weights. Generating our PCA transformed dataset and plotting PC1 and PC2 reflected what we saw with our loading factors. More information was stored in PC1, as it had a visibly higher spread than PC2.



Our scree plot showed two elbows, one at PC 1 and another at PC 6. However, only taking PC 1 and 2 results in less than 50% of the variance being explained, so it makes more sense to take PC 1-6. Overall, it seems that PCA failed to group variables together as we would like. We started with 7 numerical variables, and if we would like at least 90% of the variability explained, we would have to take 6 PCs. So PCA didn't help in reducing the dimensionality in any significant way. Therefore, it made more sense to just use the original 7 variables, rather than apply PCA.

Overall, PCA was the better choice for our dataset, because the main problem we were addressing was attempting to predict exam scores with the remaining variables as predictors. PCA, if it had been effective, would've allowed us to reduce the dimensionality of our dataset and create "summary indices" by grouping predictor variables together. Using the PCs would have resulted in a model where it is easier to explain the importance of the numerous predictor variables in explaining exam scores.

The lack of effectiveness of PCA for our task agrees with our findings from the main linear regression task, where we discovered that a model with all 9 features chosen from forward feature selection was the best at predicting exam scores compared to models with just the few features which were most highly correlated with Exam Score. The model with all 9 features does not overfit the data and enables us to easily determine feature importance, so PCA is simply not necessary. This reminds us of polygenic gene expression where many genes contribute just a bit to the variance, causing PCA to be ineffective. That seems to be the case for our dataset, where each of the 9 chosen features are contributing slightly to the overall Exam Score. Despite Attendance and Hours Studied dominating the remaining 7 features, excluding the other 7 features significantly degraded the performance of linear regression, as those other features were each still making small contributions which added up.

We chose not to apply clustering to our dataset because it did not address the problem we were trying to solve. Knowing which rows of the dataset were similar to each other wouldn't necessarily help us in predicting exam scores.

VII. Explain how your project attempted to use a neural network on the data and the results of that attempt.

Data preprocessing is similar to that of previous methods. Rows with missing values were dropped, categorical variables were encoded numerically, a subset of predictor variables were selected, and features were scaled. The data was split into 20% test and 80% training data, and the batch size was chosen to be 32, since the total number of observations was 6607. The structure of our deep neural network consisted of an input layer with 9 nodes, one for each predictor variable, two hidden layers with 64 and 32 nodes respectively, and a single output node, corresponding to predicted exam score. Stochastic gradient descent was used to update the model parameters, and MSE was chosen as our poss function. The learning rate was selected using an algorithm detailed in the next section about hyperparameter tuning. Finally, our training loop ran for 100 epochs.

To evaluate the results of our neural network, the main metric we used was mean squared error, since our response variable is numerical, and MSE was the loss function used to train our model. Also, since we used MSE as a metric to evaluate our linear regression model, this allows us to compare that model with our neural network. After the model was trained, we got the predictions from the model, computing the rMSE from the predicted and true values. We also computed MAE, R2, and MAD scores, listed below.

The results are as follows, rMSE = 2.19, MAE = 0.91, MAD = 0.62, R2 = 0.69. Based on these values, we observed that the performance of our neural network was quite similar to that of our linear regression model. In order to more fairly compare the effectiveness of the two algorithms, we tried applying techniques such as batch normalization, dropout, norm penalties, adding extra layers, and using a different optimizer (Adam instead of SGD) to the neural network. With every technique we tried, we also recomputed the optimal learning rate by recreating the plot of learning rate vs. loss. However, no matter which technique we tried, the performance of the neural network would not improve in a meaningful way compared to the linear regression model, with an R^2 score ranging from 69-71% which matches the R^2 of the linear regression model's performance on the test set and k-fold cross-validation.

Another technique we had used to compare the two models more fairly was to use the same 9 features for the neural network as we used in linear regression rather than letting the neural network use all of the features in the dataset. We thought this would improve the neural network's performance, but we were mistaken, as the R^2 actually decreased to about 62%. After conducting some research, we realized this was because of two factors: 1) neural networks can be used to model nonlinear relationships. The 9 features for linear regression were selected because they had linear relationships with the response variable, but the neural network was able to model nonlinear relationships as well, so constraining it to the same 9 features was unnecessary and 2) neural networks perform implicit feature selection by updating weights to 0 for features which do not contribute as much. The code we used to fairly compare the two models is in the "Linear Regression vs. Neural Network Comparison" notebook.

The justification to choose linear regression over neural networks as our main method for the regression task was that linear regression models allow for significantly better model interpretation and determining feature importance. By standardizing a dataset, feature importance can be compared in a linear regression model by comparing coefficients, while with a neural network, a far more computationally expensive and less straightforward approach is needed in the form of Shapley values.

VIII. **Give examples of hyperparameter tuning that you applied in preparing your project and how you chose the best parameters for models.** Hyperparameter tuning for ridge and lasso regression are described in the main report.

For our neural network, the learning rate was determined by applying an algorithm. Starting with a learning rate of 10^-5, for each iteration of training, we increased the learning rate by multiplying it with a constant, ending at a learning rate of

10 by the end of the training loop. At each step, we recorded the learning rate and loss (MSE). After this, we made a plot of the loss vs. the learning rate to see which learning rate caused the loss to decrease before increasing sharply. Based on that, we settled on a learning rate of 0.13804, which had the best results in terms of getting the algorithm to converge at a relatively low loss. This approach allowed us to avoid learning rates which were too small, causing slow convergence, or too large, causing oscillations and overshooting of the optima.