Predicting Student Performance with Educational Data

PHP 2601 (Linear Models) Final Project

Daniel Posmik, Aristofanis Rontogiannis, Jizhou Tian

2024-12-18

Table of contents

Introduction

For this project, we will be analyzing educational data. We are interested in understanding the predictors of student performance as measured by exam scores. We will be using a publicly available dataset from Kaggle that contains information about students and their exam scores.

Table 1: Variable Summary for the Educational Data

Variable Name	Variable Type	Description
Hours_Studied	numeric	Hours Studied
Attendance	numeric	Attendance
Parental_Involvement	factor	Parental Involvement
Access_to_Resources	factor	Access to Resources
Extracurricular_Activities	factor	Extracurricular Activities
Sleep_Hours	numeric	Sleep Hours
Previous_Scores	numeric	Previous Scores
Motivation_Level	factor	Motivation Level
Internet_Access	factor	Internet Access
Tutoring_Sessions	numeric	Tutoring Sessions
Family_Income	factor	Family Income
Teacher_Quality	factor	Teacher Quality
School_Type	factor	School Type
Peer_Influence	factor	Peer Influence
Physical_Activity	numeric	Physical Activity
Learning_Disabilities	factor	Learning Disability
Parental_Education_Level	factor	Parental Education Level
Distance_from_Home	factor	Distance from Home
Gender	factor	Gender
Exam_Score	numeric	Exam Score

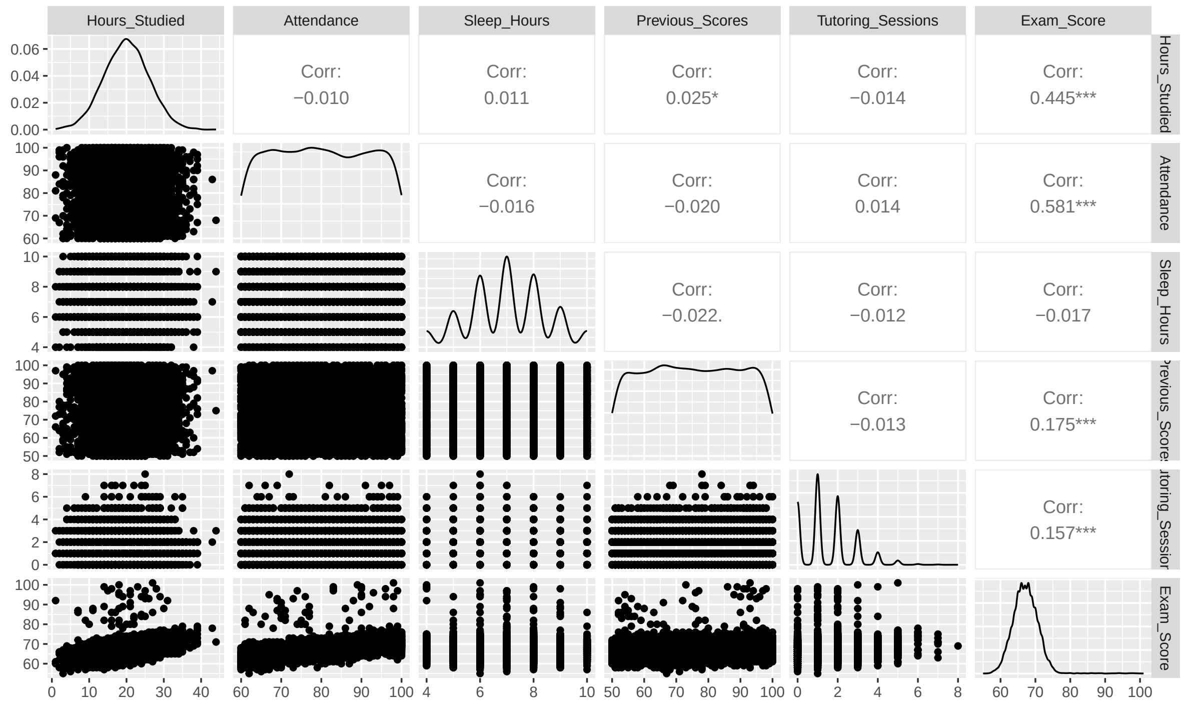
Now, we want to further explore a specific hypothesis about a subset of predictor variables. Suppose we maintain that the following variables are significant predictors:

- Hours Studied
- Attendance
- Sleep Hours
- Previous Scores
- Tutoring Sessions

We can formalize this question as follows:

- $H_0 : \left[1_{[0,\cdots,p+1]}, \quad 0_{[p+2,\cdots,P]}\right] \cdot \left[\beta_0 \quad \cdots \quad \beta_P\right]^T = \beta_0 + \cdots + \beta_{p+1} = 0$
- $H_A : \{\beta_1 \neq 0\} \cap \cdots \cap \{\beta_5 \neq 0\}$

Before we begin our analysis, let us take a look at the dependencies across these data:



Part 1: Linear Regression Analysis (Daniel)

The Least Squares Estimators

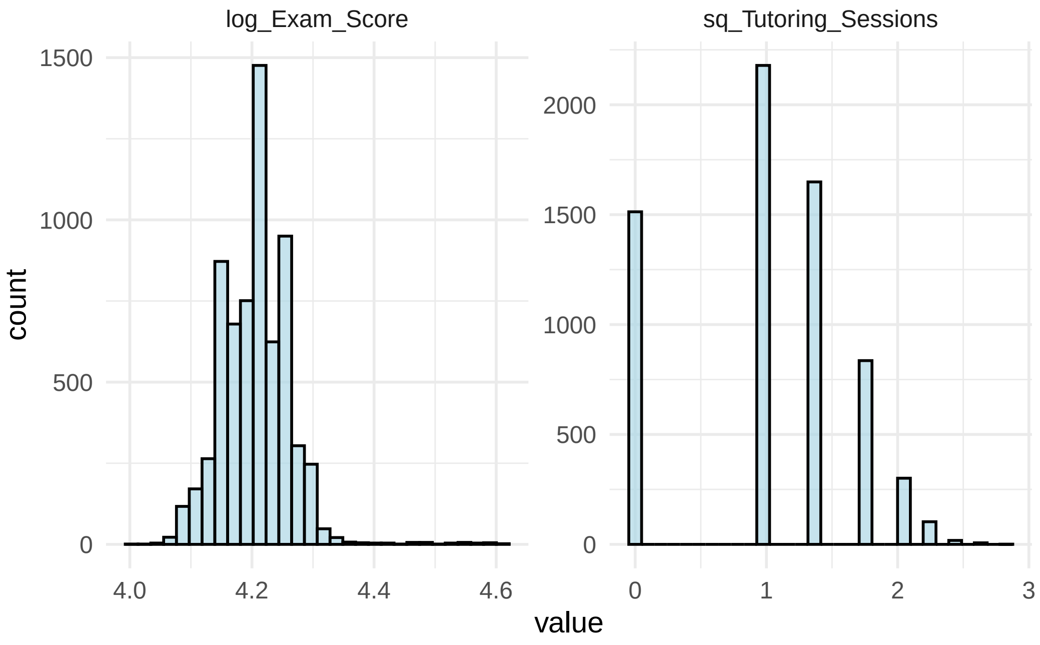
Let us begin by discussing the assumptions of linear regression model. In a Gauss-Markov setting, we assume that our linear model is of the form:

$$\mathbf{Y} = \mathbf{X}\beta + \epsilon, \quad \epsilon \sim N(0, \sigma^2 I)$$

where $\mathbb{E}[\epsilon] = 0$ and $\text{Var}[\epsilon] = \sigma^2 I$ denote the zero-mean and constant variance assumptions. In our case, we begin with $p = 5$, i.e. our design matrix has $p + 1$ columns, accounting for the intercept term. Then, we can write the model as matrices:

That being said, what the Gauss-Markov model boasts in theoretical simplicity, it often lacks in practical validity. If we refer to the exploratory analysis above, we can see that the assumptions may not hold. For one, we have one predictor variable, `Tutoring_Sessions`, and our dependent variable, `Exam_Score`, that are right-skewed. This violates the assumption of normality. Moreover, the Gauss-Markov model assumes constant variance with zeros on the off-diagonal elements of the covariance matrix. In practice, this is an assumption that is frequently violated. Interestingly, in our case the correlation between our predictor variables is indeed close to 0. If we had more substantial correlations on the off-diagonal elements, we could have solved our estimation problem with the generalized least squares estimator.

In our case, we will rememedy the normality assumption by transforming our data. We will use logaritmic transformations on the Exam score variable to achieve a normal distribution (`log_Exam_Score`) and a square root transformation of the Tutoring sessions variable to achieve a distribution that more closely resembles a normal distribution (`log_Tutoring_Sessions`). We chose the square root transformation for the tutoring sessions variable because it contains a lot of 0s, making the logarithmic transformation less suitable.



$$Y = \begin{bmatrix} Y_1 \\ Y_2 \\ \vdots \\ Y_n \end{bmatrix} = \begin{bmatrix} 1 & X_{12} & X_{13} & \cdots & X_{1(p+1)} \\ 1 & X_{22} & X_{23} & \cdots & X_{2(p+1)} \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ 1 & X_{n2} & X_{n3} & \cdots & X_{n(p+1)} \end{bmatrix} \begin{bmatrix} \beta_0 \\ \beta_1 \\ \beta_2 \\ \vdots \\ \beta_p \end{bmatrix} + \begin{bmatrix} \epsilon_1 \\ \epsilon_2 \\ \vdots \\ \epsilon_n \end{bmatrix}$$

Note that we are using the generalized matrix inverse in case the design matrix is not of full rank. The canonical matrix inverse of the form X^{-1} exists iff X is of full rank. Next, we can solve for $\widehat{\beta}$ via the normal equations:

$$
\begin{aligned}
\hat{\beta} =& (X^T X)^g X^T Y \\
=& \left(\begin{bmatrix} 1 & 1 & \cdots & 1 \\ X_{12} & X_{22} & \cdots & X_{n2} \\ \vdots & \vdots & \ddots & \vdots \\ X_{1(p+1)} & X_{2(p+1)} & \cdots & X_{n(p+1)} \end{bmatrix} \begin{bmatrix} 1 & X_{12} & \cdots & X_{1(p+1)} \\ 1 & X_{22} & \cdots & X_{2(p+1)} \\ \vdots & \vdots & \ddots & \vdots \\ 1 & X_{n2} & \cdots & X_{n(p+1)} \end{bmatrix} \right)^g \cdot \\
& \begin{bmatrix} 1 & 1 & \cdots & 1 \\ X_{12} & X_{22} & \cdots & X_{n2} \\ \vdots & \vdots & \ddots & \vdots \\ X_{1(p+1)} & X_{2(p+1)} & \cdots & X_{n(p+1)} \end{bmatrix} \begin{bmatrix} Y_1 \\ Y_2 \\ \vdots \\ Y_n \end{bmatrix}
\end{aligned}
$$

Using the R `lm()` function, we can estimate the coefficients of the linear model:

```
Call:
lm(formula = log_Exam_Score ~ Hours_Studied + Attendance + Sleep_Hours +
    Previous_Scores + sq_Tutoring_Sessions, data = educ_dta)

Residuals:
     Min       1Q   Median       3Q      Max
-0.08391 -0.01675 -0.00173  0.01336  0.37210

Coefficients:
                       Estimate Std. Error t value Pr(>|t|)
(Intercept)           3.818e+00  4.460e-03 856.076   <2e-16 ***
Hours_Studied         4.350e-03  6.916e-05  62.899   <2e-16 ***
Attendance            2.951e-03  3.588e-05  82.256   <2e-16 ***
Sleep_Hours          -2.646e-04  2.822e-04  -0.938    0.348
Previous_Scores       7.146e-04  2.878e-05  24.827   <2e-16 ***
sq_Tutoring_Sessions  1.339e-02  6.417e-04  20.865   <2e-16 ***
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

Residual standard error: 0.03366 on 6601 degrees of freedom
Multiple R-squared:  0.6387,	Adjusted R-squared:  0.6385
F-statistic:  2334 on 5 and 6601 DF,  p-value: < 2.2e-16
```

We can see that all variables except for `Sleep_Hours` are significant predictors of exam scores, even at a 1% significance level. So what does this tell us about our hypothesis? We will further examine this question in the next subsection.

	(1)
(Intercept)	3.818***
	(0.004)
Hours_Studied	0.004***
	(0.000)
Attendance	0.003***
	(0.000)
Sleep_Hours	0.000
	(0.000)
Previous_Scores	0.001***
	(0.000)
sq_Tutoring_Sessions	0.013***
	(0.001)
Num.Obs.	6607
R2	0.639
R2 Adj.	0.638
AIC	−26 058.3
BIC	−26 010.7
Log.Lik.	13 036.154
F	2334.162
RMSE	0.03

+ $p < 0.1$, * $p < 0.05$, ** $p < 0.01$, *** $p < 0.001$

Hypothesis Testing

Our estimation question is a hypothesis testing problem. In the following, we will rigorously treat is such, testing whether our subset of predictors (see above) is jointly significant in the prediction of exam scores. Before we proceed, let us introduce additional notation in our hypothesis testing problem:

$$\mathbf{K}^T\beta = \begin{bmatrix} 1_{[0,\cdots,p+1]}, & 0_{[p+2,\cdots,P]} \end{bmatrix} \cdot \begin{bmatrix} \beta_0 & \cdots & \beta_P \end{bmatrix}^T = \beta_0 + \cdots + \beta_{p+1} = \mathbf{M}_{1,(p+1)}$$

where $\{\beta_0, \ldots, \beta_{p+1}\}$ are the coefficients of the predictors we are interested in and $\{\beta_{p+2}, \ldots, \beta_P\}$ are the coefficients of the remaining predictors. Naturally, $p \leq P$.

A necessary condition for the hypothesis to be testable is that $\mathbf{K}^T\beta$ is estimable. We say $\exists\ A$ s.t. $X^T A = K^T$, i.e. the rows of K are linearly dependent on the rows of X. Indeed, we can verify this without the calculation because we can see that $\mathbf{L}^T$ can be expressed as a linear combination of the design matrix, i.e. the columns space of X, $\mathbb{C}(X)$, contains the column space of K, $\mathbb{C}(\mathbf{K})$. A counterexample would be if one of our predictors consisted of 0's only, rendering us unable to estimate $\mathbf{K}^T$ with $\mathbf{X}$.

We are now ready to state an important intermediate distributional result. Since $\mathbf{K}^T\beta$ is estimable, its best linear unbiased estimator (BLUE) is given by:

$$\begin{aligned} \mathbf{K_i}^T\hat{\beta} &\sim N(\mathbf{K_i}^T(X^TX)^g X^TX\beta, \sigma^2\mathbf{K_i}^T(X^TX)^g\mathbf{K_i}) \quad \text{and} \\ \mathbf{K}^T\hat{\beta} &\sim N(\mathbf{K}^T(X^TX)^g X^TX\beta, \sigma^2\mathbf{K}^T(X^TX)^g\mathbf{K}) \end{aligned}$$

Indeed, we can can test our hypothesis by constructing a quadratic form. While this is certainly not the only way to test our hypothesis, it is a tractable method to incorporate the precision of each $\hat{\beta}_i$ into our hypothesis testing framework. We will see momentarily that this quadratic form results in favorable distributional properties thanks to the previous normal distributional result. Now, defining $H := K(X^TX)^g K^T$,

$$\begin{aligned} &\mathbf{K}^T\hat{\beta} \sim N(\mathbf{K}^T(X^TX)^g X^TX\beta, \sigma^2\mathbf{K}^T(X^TX)^g\mathbf{K}) \\ \iff\ &\mathbf{K}^T\hat{\beta} \sim N(\mathbf{K}\beta, \sigma^2 H) \end{aligned}$$

we can construct the quadratic form

$$(K\beta)^T(\sigma^2 H)^{-1}(K\hat{\beta}) \sim \chi^2_{\text{df}=\text{rank}(H)}(\lambda)$$

where the non-centrality parameter $\lambda = \frac{1}{2}(K\beta)^T(\sigma^2 H)^{-1}(K\beta)$ by the well-known distributional result of a normal quadratic form. We are now ready to construct the F-test statistic as follows:

$$F := \frac{\left((K\beta)^T(\sigma^2 H)^{-1}(K\beta)\right)/\text{rank}(H)}{\text{RSS}/(n-p)} \sim \frac{\chi^2(\lambda)}{\chi^2} \sim F_{\text{rank}(H),n-p}(\lambda)$$

We have successfully constructed a statistical test that allows us to test our hypothesis with a simple F-test. This is very attractive seeing how this test incorportates the precision of our estimates into the hypothesis testing framework, yet is computationally simple. In `R`, we can use the `anova()` function to perform this test.

```
Analysis of Variance Table

Model 1: log_Exam_Score ~ 1
Model 2: log_Exam_Score ~ Hours_Studied + Attendance + Sleep_Hours + Previous_Scores +
    sq_Tutoring_Sessions
  Res.Df     RSS Df Sum of Sq      F    Pr(>F)
1   6606 20.6969
2   6601  7.4771  5     13.22 2334.2 < 2.2e-16 ***
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

The result shows that under the null hypothesis, the probability of getting a more extreme result than our calculate F-test statistics `Pr(>F)` is $2.2e-16$. This evidence would lead us to reject the null hypothesis and conclude that our subset of predictors is indeed a significant predictor of exam scores. The observed test statistics agree exactly with the ones reported in the regression table. Using the F-statistic is important in settings like ours when we are interested in joint model significance rather than individual predictor significance. It is noteworthy that this analysis could certainly be extended to different model specifications, however, this is beyond the scope of this project.

Part 2: Linear Regression Analysis - Elastic Net (Aristofanis)

In our analysis, we initially explored dimensionality reduction techniques such as PCA, but the results were not promising due to the structure of our dataset, where 13 out of 20 variables are categorical. When we limited the analysis to the continuous variables, PCA required 5 out of 7 variables to explain a satisfactory proportion of variance, which limited its utility. Consequently, we decided to shift our approach to a regularization technique, specifically Elastic Net, which allows us to incorporate the full dataset, including categorical and continuous variables. This method not only identifies important variables but also enables us to evaluate how these results align with the findings from linear regression using only the continuous variables.

At this point, contrary to our previous analysis, we will use the whole dataset to apply Elastic Net Regression- a regularization technique that balances feature selection and model complexity. We are aiming to achieve robust predictions while addressing multicollinearity and sparsity in the data."

Elastic Net Regularization

Elastic Net is a regularized regression technique that combines the strengths of two other methods: Ridge Regression and Lasso Regression. It is especially useful when you have a dataset with many predictors, some of which may be highly correlated.

Key Features of Elastic Net:

Combination of Ridge and Lasso:

- **Ridge Regression:** Adds an L_2-norm penalty to the loss function, which helps handle multicollinearity but does not perform variable selection.
- **Lasso Regression:** Adds an L_1-norm penalty, which can shrink some coefficients to exactly zero, effectively performing variable selection.
- **Elastic Net** combines both penalties, controlled by a mixing parameter (α).

The penalty term in Elastic Net is:

$$\lambda \left(\alpha \|\beta\|_1 + (1-\alpha) \frac{1}{2} \|\beta\|_2^2 \right)$$

where:

- α: Controls the balance between L_1 (Lasso) and L_2 (Ridge) penalties.
- λ: Overall regularization strength.

How Elastic Net Handles Key Challenges:

Handles Multicollinearity:

When predictors are highly correlated, Lasso may arbitrarily select one variable. Elastic Net tends to share the weight across correlated predictors.

Variable Selection:

Like Lasso, Elastic Net can shrink some coefficients to zero, effectively removing those predictors from the model.

Elastic Net with continuous outcome

Model Performance and Results

The Elastic Net regression model was applied to predict the `exam score`, achieving an R^2 value of approximately 0.76. This indicates that the model explains 76% of the variance in exam scores, demonstrating a strong fit and reliable performance.

The table of coefficients highlights the key predictors and their influence on exam scores. Negative coefficients, such as `Access_to_ResourcesLow` (-2.0123), `Teacher_QualityLow` (-1.1046), and `Motivation_LevelLow` (-1.0276), indicate that limited resources, low teacher quality, and low motivation significantly decrease exam scores. In contrast, positive coefficients like `Parental_InvolvementHigh` (1.0873), `Internet_AccessYes` (1.0020), and `Distance_from_HomeNear` (0.8904) show that high parental involvement, internet access, and proximity to home positively impact scores.

Continuous variables, such as `Hours_Studied` (0.2901) and `Physical_Activity` (0.1636), further contribute positively to exam scores, while predictors like `Sleep_Hours` (-0.0187) and `School_TypePublic` (-0.0167) show minimal effects.

Elastic Net proved particularly useful for handling the dataset's structure, which includes a mix of categorical and continuous variables. Unlike PCA, which struggled due to the dominance of categorical variables (13 out of 20), Elastic Net enabled the inclusion of all predictors, improving performance and variable selection. This approach not only identified key factors influencing exam scores but also allowed for comparisons with simpler models using only continuous predictors.

In summary, Elastic Net successfully balanced variable selection and prediction accuracy, providing a comprehensive understanding of the most significant factors driving exam scores.

Table 2: Elastic Net Coefficients

Predictor	Elastic Net Coefficient
(Intercept)	40.7199
Access_to_ResourcesLow	-2.0123
Teacher_QualityLow	-1.1046
Family_IncomeLow	-1.0943
Parental_InvolvementHigh	1.0873
Motivation_LevelLow	-1.0276
Peer_InfluencePositive	1.0049
Internet_AccessYes	1.0020
Access_to_ResourcesMedium	-0.9558
Distance_from_HomeNear	0.8904
Parental_InvolvementLow	-0.8389
Learning_DisabilitiesYes	-0.7820
Family_IncomeMedium	-0.6332
Extracurricular_ActivitiesYes	0.6087
Teacher_QualityMedium	-0.5468
Tutoring_Sessions	0.5074
Peer_InfluenceNeutral	0.5034
Parental_Education_LevelHigh School	-0.4799
Motivation_LevelMedium	-0.4478
Parental_Education_LevelPostgraduate	0.4384
Distance_from_HomeModerate	0.3105
Hours_Studied	0.2901
Attendance	0.1997
Physical_Activity	0.1636
GenderMale	-0.0565
Previous_Scores	0.0510
Sleep_Hours	-0.0187
School_TypePublic	-0.0167

Note: Elastic Net (or Lasso) regularization applies penalties to the model coefficients to encourage sparsity, meaning some predictors may be dropped if their coefficients shrink to zero. However, in this case, Elastic Net did not drop any variables because all predictors contributed to reducing the loss function. Even with the regularization applied, the inclusion of all variables suggests that each predictor holds some explanatory power for the outcome variable (exam scores).

The mix of categorical and continuous variables, along with their relationships to the target variable, likely prevents any single predictor from being entirely irrelevant. This highlights the flexibility of Elastic Net, as it balances between Lasso (L1 penalty, encouraging sparsity) and

Ridge regression (L2 penalty, shrinking coefficients without elimination), ensuring that even small but relevant predictors remain in the model.

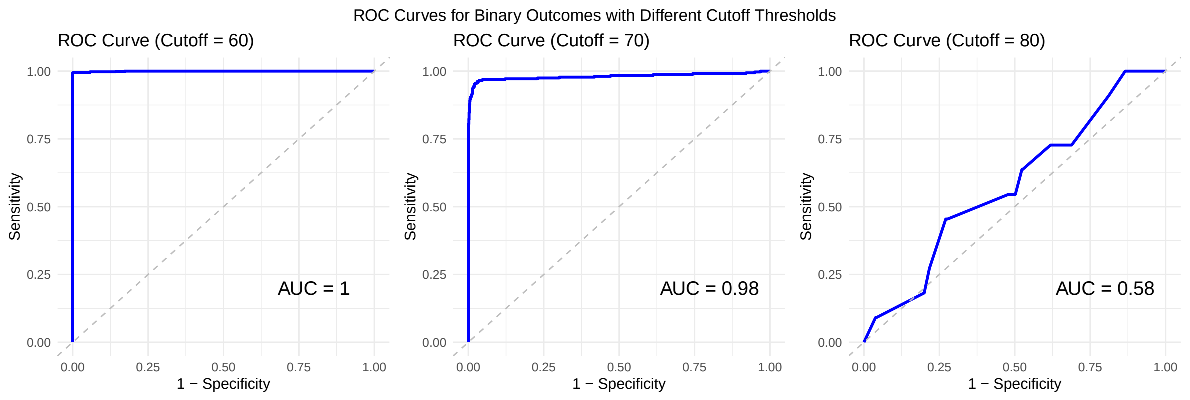
Elastic Net with categorical outcome

ROC Curve Analysis for Binary Exam Score Thresholds

To evaluate the model's performance with a binary exam score outcome, three thresholds (60, 70, and 80) were tested. At a cutoff of 60, the model achieved perfect classification with an AUC of 1, as most students scored above this threshold. For a cutoff of 70, the AUC was 0.98, reflecting excellent performance. This threshold aligns closely with the dataset's structure, where the median and mean scores are near the 3rd quantile (69). As a result, predicting whether a student's score falls within or outside the top 25% can be done almost perfectly.

At a cutoff of 80, the AUC dropped to 0.58, indicating the model's ability to distinguish between the classes was only slightly better than random guessing. This performance decline suggests that class imbalance and reduced class separation make it harder to classify scores at higher thresholds.

In summary, the model performs exceptionally well at lower thresholds (60 and 70), benefitting from the dataset's imbalanced nature, but struggles at higher thresholds like 80 due to increased classification difficulty.



Part 3: Non-linear Regression Analysis, Random Forest (Jizhou)

Random Forest is a powerful and flexible ensemble learning method used for regression, classification, and other tasks. It is based on the idea of combining multiple decision trees to improve predictive performance and robustness, aiming to overcome the over-fitting problem of individual decision tree.

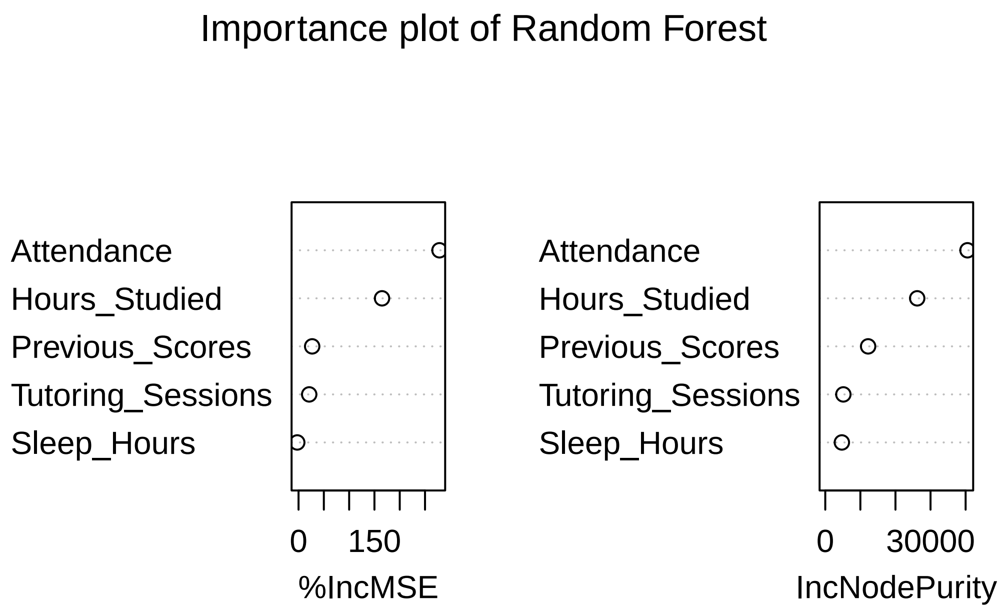
Each decision tree in the Random Forest is trained on a bootstrap sample of the original data, meaning that each tree is trained using approximately two-thirds (63.2%) of the entire training data set. Furthermore, at each split in a tree, a subset of the predictor variables is selected **randomly** to determine the split, further reducing overfitting and encouraging model variance. This process ensures that the individual trees in the ensemble are decorrelated, making the model robust to noise and capable of generalizing well.

One of the key advantages of Random Forest is its ability to handle high-dimensional data and complex non-linear relationships between variables. It is also less prone to overfitting compared to individual decision trees, due to the averaging effect across multiple trees. Additionally, Random Forest provides measures of variable importance, enabling insights into the contribution of predictors to the model.

In this section, we will utilize Random Forest to analyze the data set and capture complex non-linear relationships between the variables while assessing its predictive performance.

Random Forest Model

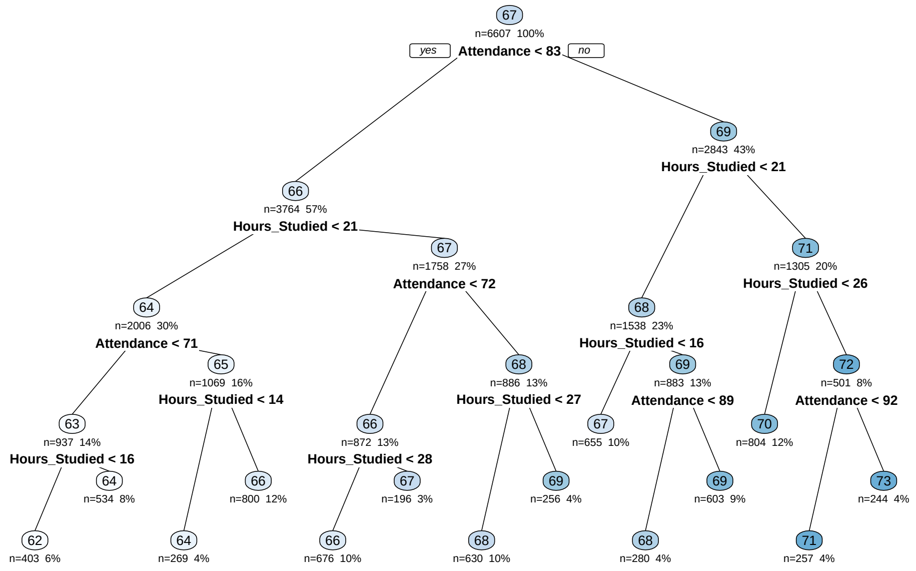
We construct a random forest model with 1,000 trees, 2 variables randomly sampled as a candidate at each split, and a minimum terminal node size of 5. The model's out-of-bag (OOB) MSE is 6.638. From the importance plot, we observe that Attendance is the most important variable, while Sleep_Hours is the least important. This aligns with our findings from the linear regression analysis, where Sleep_Hours is found to be insignificant.



Approximation by a Single Regression Tree

Despite its strengths, Random Forest is computationally intensive, especially when applied to large data sets with a large number of trees. Moreover, it can be less interpretable compared to simpler models, as the ensemble structure makes it challenging to directly understand the decision-making process.

To make the Random Forest more interpretable, we fit a single regression tree to approximate and visualize the results obtained from the Random Forest. The predicted response values from the Random Forest based on OOB samples are used as the response variable, and the same covariates from the Random Forest are employed to fit the regression tree. To assess how well the regression tree approximates the Random Forest, we calculate the Pearson correlation between the predicted response values from the Random Forest and those from the single regression tree.



The correlation between the Random Forest and the single regression tree achieves 0.898, indicating that the single tree provides a good approximation of the Random Forest. This regression tree highlights the factors influencing the target outcome exam scores, with **Attendance** emerging as the most critical predictor. The tree's root splits data based on whether Attendance is below or above 83, reflecting its overall importance. For lower Attendance values, further splits are influenced by **Hours_Studied**. Both lower Attendance and fewer Hours_Studied lead to lower exam scores, aligning with expectations. The tree reveals **interaction effects** between **Attendance** and **Hours_Studied**, as their thresholds creating different groups with varying predicted values.

Part 4: Non-linear Regression Analysis, Gradient Boosting Algorithm (Aristofanis)

Gradient Boosting Machines (GBM) - Gradient Boosting Algorithm

Overview

Gradient Boosting Machines (GBM) is a powerful ensemble learning technique used for both regression and classification problems. It sequentially combines multiple weak learners—typically decision trees—to build a strong predictive model. GBM optimizes predictive performance by minimizing a specified loss function through gradient descent, iteratively correcting errors from previous models.

Definition and Purpose

- **Definition**: Gradient boosting is an ensemble method that sequentially trains weak learners (e.g., decision trees) to predict the residuals or errors of prior models. By combining these weak learners, GBM produces a robust and accurate final model.

- **Purpose**: GBM aims to reduce prediction errors by optimizing model weights iteratively. At each step, it focuses on the mistakes of the previous models, gradually improving accuracy.

How GBM Works

1. **Step 1**: Start with an initial prediction, often a simple baseline model (e.g., the mean value for regression).

2. **Step 2**: Compute the residuals (errors) between the actual and predicted values.

3. **Step 3**: Train a weak learner, such as a shallow decision tree, to predict the residuals.

4. **Step 4**: Add the new learner to the ensemble and update the overall prediction.

5. **Step 5**: Repeat this process until the model converges or a predefined number of iterations is reached.

Each weak learner corrects the errors of its predecessors, resulting in a final model that minimizes the loss function effectively.

Key Features of GBM

- **Strengths**:
  - Handles non-linear relationships well.
  - Provides high predictive accuracy.
  - Can work with both continuous and categorical predictors.
- **Challenges**:
  - Prone to overfitting if not carefully tuned (e.g., with parameters like learning rate, tree depth, and number of iterations).
  - Computationally intensive for large datasets.

To achieve optimal performance, GBM models rely on key hyperparameters:

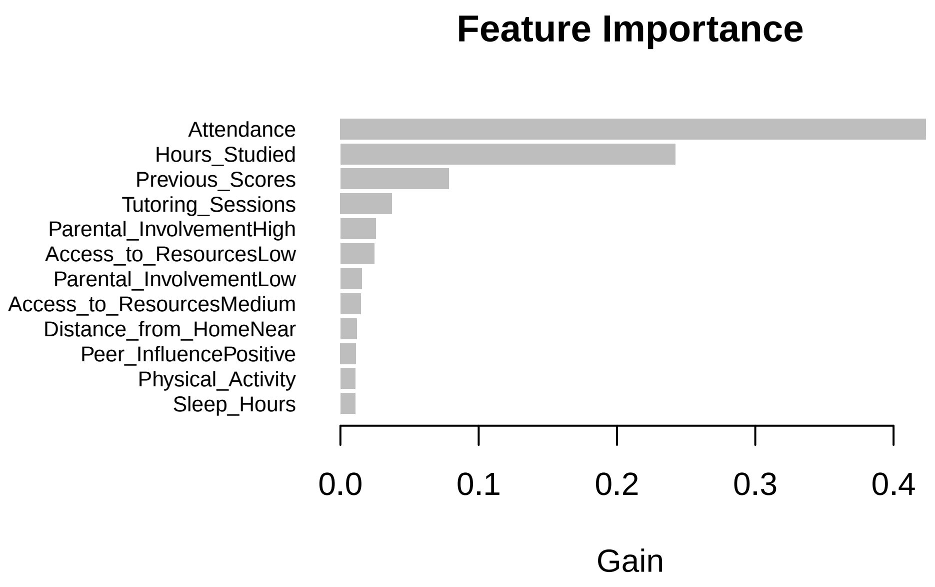
- **Learning Rate (Shrinkage)**: Controls the contribution of each weak learner. Smaller values improve generalization but require more iterations.
- **Number of Trees**: The total number of weak learners to combine.
- **Tree Depth**: The depth of each decision tree, balancing model complexity.
- **Loss Function**: Defines the error to minimize, such as Mean Squared Error (MSE) for regression.

Compared to simpler linear models like regression or Lasso, GBM excels in capturing complex, non-linear interactions within the data. It is robust to multicollinearity and can handle missing values effectively, making it a versatile choice for predictive modeling tasks.

The Gradient Boosting Machine (GBM) was applied to the dataset to predict the exam scores, leveraging both categorical and continuous variables. The results demonstrate the model's ability to capture relationships in the data effectively, producing strong predictive performance.

Feature	Gain	Cover	Frequency
Attendance	0.4234050	0.1493851	0.1261209
Hours_Studied	0.2422001	0.1547481	0.1483431
Previous_Scores	0.0782877	0.0875713	0.1423002
Tutoring_Sessions	0.0373611	0.0662183	0.0730994
Parental_InvolvementHigh	0.0255775	0.0339374	0.0309942
Access_to_ResourcesLow	0.0244406	0.0486528	0.0319688

Feature	Gain	Cover	Frequency
Parental_InvolvementLow	0.0155172	0.0279404	0.0309942
Access_to_ResourcesMedium	0.0147545	0.0280198	0.0272904
Distance_from_HomeNear	0.0116891	0.0326773	0.0255361
Peer_InfluencePositive	0.0113251	0.0341277	0.0239766



GBM Results and Interpretation

The Gradient Boosting Machine (GBM) was applied to predict exam scores, utilizing both continuous and categorical variables. The model demonstrates strong predictive performance, with results highlighting the relative importance of key features in determining student outcomes. The most influential predictor is `Attendance`, with a gain of 0.42, indicating that regular class participation is the strongest driver of exam performance. `Hours_Studied` follows closely with a gain of 0.24, underscoring the critical role of study time in improving scores. `Previous_Scores` ranks third (gain = 0.078), showing that prior academic achievements significantly influence current performance. Additional features, such as `Tutoring_Sessions`, `Parental_InvolvementHigh`, and `Access_to_ResourcesLow`, contribute moderately, emphasizing the impact of external academic support, family involvement, and access to resources. Features like `Peer_InfluencePositive` and `Distance_from_HomeNear` have smaller contributions but still indicate subtle relationships with exam scores.

The feature importance plot further illustrates the dominance of `Attendance` and `Hours_Studied`, with a steep drop in importance among other variables. This suggests that while several factors influence performance, regular attendance and dedicated study time are the most consistent and impactful predictors.

Note: The results from the GBM are **very similar to the Random Forest model** presented in the previous section, where only continuous variables were used. This consistency suggests that both ensemble techniques—while leveraging different learning mechanisms—are effective at identifying important predictors and delivering robust performance.

Conclusion

In this project, we explored various statistical and machine learning techniques to predict student exam scores using educational data. Our analysis began with linear regression, which provided a foundational understanding of the relationships between the predictors and the outcome. Key predictors such as `Hours_Studied`, `Attendance`, and `Previous_Scores` emerged as significant contributors to exam performance, with the model achieving a good fit.

To address multicollinearity and include both continuous and categorical variables, we applied Elastic Net regression, which demonstrated strong performance with an R^2 of approximately 0.76. Elastic Net successfully balanced variable selection and prediction accuracy, identifying important predictors like `Access_to_Resources`, `Teacher_Quality`, and `Parental_Involvement` while retaining all features due to their contributions to reducing the loss function. ROC analysis with binary exam outcomes further showed that classification accuracy was near perfect for thresholds like 60 and 70 but declined for higher thresholds, reflecting dataset imbalances.

In the non-linear analysis, Random Forest and Gradient Boosting Machines (GBM) were implemented to capture complex, non-linear relationships in the data. Both models highlighted `Attendance` and `Hours_Studied` as the most critical predictors, aligning with the results from Elastic Net and linear regression. The GBM model provided a detailed feature importance analysis, with gains of 0.42 for `Attendance` and 0.24 for `Hours_Studied`, reinforcing their significance. Notably, the results from GBM were very similar to those of Random Forest, further validating the robustness of the identified predictors and the models' reliability.

Overall, our findings consistently emphasize the importance of class participation, study habits, and previous academic performance in predicting student exam scores. While linear models provided interpretability and simplicity, the non-linear models (Random Forest and GBM) offered higher flexibility and accuracy, making them valuable tools for understanding and predicting student outcomes. This comprehensive analysis highlights the strengths of different approaches and underscores the utility of ensemble techniques in educational data modeling.

Code Appendix

```r
# Set up knit environment
knitr::opts_chunk$set(echo = F)
knitr::opts_chunk$set(error = F)
knitr::opts_chunk$set(warning = F)
knitr::opts_chunk$set(message = F)

# Load necessary packages
library(tidyverse)
library(kableExtra)
library(knitr)
library(broom)
library(ggplot2)
library(naniar)
library(gtsummary)
library(GGally)
library(MASS)
library(modelsummary)
library(randomForest)
library(rpart.plot)

# Load the data
educ_dta <- read_csv("student_performance.csv") %>%
  mutate(
    Parental_Involvement = as.factor(Parental_Involvement),
    Access_to_Resources = as.factor(Access_to_Resources),
    Extracurricular_Activities = as.factor(Extracurricular_Activities),
    Motivation_Level = as.factor(Motivation_Level),
    Internet_Access = as.factor(Internet_Access),
    Family_Income = as.factor(Family_Income),
    Teacher_Quality = as.factor(Teacher_Quality),
    School_Type = as.factor(School_Type),
    Peer_Influence = as.factor(Peer_Influence),
    Learning_Disabilities = as.factor(Learning_Disabilities),
    Parental_Education_Level = as.factor(Parental_Education_Level),
    Distance_from_Home = as.factor(Distance_from_Home),
    Gender = as.factor(Gender)
  )

# Summary table
```

```r
table_summary <- tibble(
  "Variable Name" = colnames(educ_dta),
  "Variable Type" = sapply(educ_dta, class),
  "Description" = c(
    "Hours Studied", #1
    "Attendance", #2
    "Parental Involvement", #3
    "Access to Resources", #4
    "Extracurricular Activities", #5
    "Sleep Hours", #6
    "Previous Scores", #7
    "Motivation Level", #8
    "Internet Access", #9
    "Tutoring Sessions", #10
    "Family Income", #11
    "Teacher Quality", #12
    "School Type", #13
    "Peer Influence", #14
    "Physical Activity", #15
    "Learning Disability", #16
    "Parental Education Level", #17
    "Distance from Home", #18
    "Gender", #19
    "Exam Score" #20
  )
)

# Display the table
knitr::kable(table_summary,
  caption = "Variable Summary for the Educational Data")

# Save the table
write.csv(table_summary, "table_summary.csv", row.names = F)

pred_var <- c("Hours_Studied",
              "Attendance",
              "Sleep_Hours",
              "Previous_Scores",
              "Tutoring_Sessions")

pred_dta <- educ_dta %>%
  dplyr::select(all_of(pred_var), "Exam_Score")
```

```r
p.pairs <- GGally::ggpairs(pred_dta)

p.pairs

ggsave("correlation_matrix.png", p.pairs, width = 10, height = 6)

# Log-transform skewed variables
educ_dta <- educ_dta %>%
  mutate(
    log_Exam_Score = ifelse(Exam_Score == 0, 0, log(Exam_Score)),
    sq_Tutoring_Sessions = sqrt(Tutoring_Sessions)
    )

# Display histograms next to each other
p.hist <- educ_dta %>%
  dplyr::select(log_Exam_Score, sq_Tutoring_Sessions) %>%
  gather() %>%
  ggplot(aes(value)) +
  geom_histogram(bins = 30, color = "black",
    fill = "lightblue", alpha = 0.7) +
  facet_wrap(~key, scales = "free") +
  theme_minimal()

p.hist

ggsave("histograms.png", p.hist, width = 10, height = 6)

# Fit the linear model
lm_model <-
  lm(log_Exam_Score ~ Hours_Studied + Attendance +
                      Sleep_Hours + Previous_Scores + sq_Tutoring_Sessions,
                      data = educ_dta)

# Save results
write.csv(tidy(lm_model), "lm_results.csv", row.names = F)

# Summary
summary(lm_model)

# Print with modelsummary
modelsummary::modelsummary(lm_model,
  stars = TRUE,
  caption = "Linear Regression Results")

# Null model
null_model <- lm(log_Exam_Score ~ 1, data = educ_dta)

# Perform the F-test
anova.tbl <- anova(null_model, lm_model)

anova.tbl

# Save the table
write.csv(anova.tbl, "anova_results.csv", row.names = F)

# Load the data
educdata <- read_csv("student_performance.csv") %>%
  mutate(
    Parental_Involvement = as.factor(Parental_Involvement),
    Access_to_Resources = as.factor(Access_to_Resources),
    Extracurricular_Activities = as.factor(Extracurricular_Activities),
    Motivation_Level = as.factor(Motivation_Level),
    Internet_Access = as.factor(Internet_Access),
    Family_Income = as.factor(Family_Income),
    Teacher_Quality = as.factor(Teacher_Quality),
    School_Type = as.factor(School_Type),
    Peer_Influence = as.factor(Peer_Influence),
    Learning_Disabilities = as.factor(Learning_Disabilities),
    Parental_Education_Level = as.factor(Parental_Education_Level),
    Distance_from_Home = as.factor(Distance_from_Home),
    Gender = as.factor(Gender)
  )

#View(educdata)
#str(educdata)
# The percentage of missingness is very low (way less than 5%), so we can delete the rows wi
data <- na.omit(educdata)
s<-summary(data) # the mean is always very close to the median in every continuous variable,
# Check the new dimensions
#dim(data) #6378x20

# Lets check correlations of the numeric variables
# Select only numeric columns
#numeric_data <- data[sapply(data, is.numeric)]
```

```r
# Compute correlation matrix
#cor_matrix <- cor(numeric_data, use = "complete.obs")  # Ignores missing values
#print(cor_matrix)

#linear model selection and regularization (ISLR-Tibshirani)
# Load necessary libraries
library(glmnet)
library(ggplot2)
library(dplyr)

data <- na.omit(educdata)

# Specify the target and predictors
target <- "Exam_Score"
categorical_cols <- names(data)[sapply(data, is.factor)]
numeric_cols <- names(data)[sapply(data, is.numeric) & names(data) != target]

# Create dummy variables for categorical predictors
dummies <- model.matrix(~ . - 1, data = data[categorical_cols])
X <- cbind(data[, numeric_cols], dummies)
y <- data[[target]]

# Split data into training and testing sets
set.seed(581)
train_index <- sample(1:nrow(X), 0.7 * nrow(X))
X_train <- X[train_index, ]
X_test <- X[-train_index, ]
y_train <- y[train_index]
y_test <- y[-train_index]

# Fit Elastic Net model using cv.glmnet
set.seed(581)
elastic_net_model <- cv.glmnet(
  x = as.matrix(X_train),
  y = y_train,
  family = "gaussian", # Use gaussian for continuous outcome
  alpha = 0.5,         # Equivalent to l1_ratio = 0.5
  standardize = TRUE   # Standardize predictors
)

# Extract the best lambda
best_lambda <- elastic_net_model$lambda.min
```

```r
# Predict on the test set
predicted_values <- predict(elastic_net_model, newx = as.matrix(X_test), s = best_lambda)

# Compute R-squared (or other regression metrics)
residuals <- y_test - predicted_values
rss <- sum(residuals^2)  # Residual sum of squares
tss <- sum((y_test - mean(y_test))^2)  # Total sum of squares
r_squared <- 1 - rss / tss

# Print R-squared
#cat("R-squared on test data:", round(r_squared, 4), "\n")

# Extract coefficients at best lambda
coefficients <- coef(elastic_net_model, s = best_lambda)

# Convert coefficients to a data frame
coef_table <- data.frame(
  Variable = rownames(coefficients),
  Coefficient = as.numeric(coefficients)
)

# Filter for non-zero coefficients
coef_table <- coef_table %>%
  filter(Coefficient != 0) %>%
  arrange(desc(abs(Coefficient)))  # Sort by the absolute value of coefficients

# Print the formatted table
coef_table %>%
  rename(
    Predictor = Variable,
    `Elastic Net Coefficient` = Coefficient
  ) %>%
  knitr::kable(
    caption = "Elastic Net Coefficients",
    format = "markdown",
    digits = 4
  )


# Load necessary libraries
library(glmnet)
library(pROC)
library(egg)

data <- na.omit(educdata)
# Transform Exam_Score into a binary variable and remove the original column
data <- data %>%
  mutate(Exam_Score_60 = as.factor(ifelse(Exam_Score > 60, "Pass", "Fail")),
         Exam_Score_70 = as.factor(ifelse(Exam_Score > 70, "Pass", "Fail")),
         Exam_Score_80 = as.factor(ifelse(Exam_Score > 80, "Pass", "Fail")))

# the median (and the mean) is very close to 3rd quantile. We choose threshold = 70, because

# Specify the target and predictors
targets <- c("Exam_Score_60", "Exam_Score_70", "Exam_Score_80")
cutoffs <- c("60", "70", "80")

ROC_curves = list()
for (i in 1:3) {
  target = targets[i]
  cutoff = cutoffs[i]
  categorical_cols <- names(data)[sapply(data, is.factor) &
                                    !names(data) %in% targets]
  numeric_cols <- names(data)[sapply(data, is.numeric) & names(data) != "Exam_Score"]

  # Create dummy variables for categorical predictors
  dummies <- model.matrix( ~ . - 1, data = data[categorical_cols])

  # Combine predictors and target into a single dataset
  combined_data <- cbind(data[, numeric_cols], dummies)
  combined_data$Exam_Score_binomial <- as.numeric(data[[target]]) - 1  # Convert "Pass"/"Fai

  # Split data into training and testing sets
  set.seed(581)
  train_index <- sample(1:nrow(combined_data), 0.7 * nrow(combined_data))
  X_train <- combined_data[train_index, -ncol(combined_data)]
  y_train <- combined_data[train_index, ncol(combined_data)]
  X_test <- combined_data[-train_index, -ncol(combined_data)]
  y_test <- combined_data[-train_index, ncol(combined_data)]

  # Fit Elastic Net model using cv.glmnet
  set.seed(581)
  elastic_net_model <- cv.glmnet(
    x = as.matrix(X_train),
    y = y_train,
    family = "binomial",
    # Use binomial for binary classification
    alpha = 0.5,
    # Equivalent to l1_ratio = 0.5
    standardize = TRUE,
    type.measure = "auc"  # Use AUC as the performance metric
  )

  # Extract the best lambda
  best_lambda <- elastic_net_model$lambda.min

  # Predict probabilities on the test set
  predicted_probabilities <- predict(
    elastic_net_model,
    newx = as.matrix(X_test),
    s = best_lambda,
    type = "response"
  )

  # Generate predictions
  predicted_classes <- ifelse(predicted_probabilities > 0.5, 1, 0)

  # Compute confusion matrix
  conf_matrix <- table(Predicted = predicted_classes, Actual = y_test)
  #print(conf_matrix)

  # Compute AUC and plot ROC curve
  roc_curve <- roc(y_test, as.numeric(predicted_probabilities))
  auc_value <- auc(roc_curve)

  # Plot ROC curve
  roc = ggplot(
    data.frame(
      Specificity = rev(roc_curve$specificities),
      Sensitivity = rev(roc_curve$sensitivities)
    ),
    aes(x = 1 - Specificity, y = Sensitivity)
  ) +
    geom_line(color = "blue", size = 1) +
    geom_abline(
      slope = 1,
      intercept = 0,
      linetype = "dashed",
      color = "grey"
    ) +
    annotate(
      "text",
      x = 0.8,
      y = 0.2,
      label = paste("AUC =", round(auc_value, 2)),
      size = 5,
      color = "Black"
    ) +
    labs(x = "1 - Specificity", y = "Sensitivity", title = paste0("ROC Curve (Cutoff = ", cu
    theme_minimal()

  ROC_curves[[i]] = roc
}

ROC_curves = egg::ggarrange(plots = ROC_curves,
               nrow = 1,
               top = "ROC Curves for Binary Outcomes with Different Cutoff Thresholds")
#ggsave(ROC_curves, "ROC_curves.png", width = 12, height = 4.5, unit = "in")

# Extract coefficients at best lambda
# coefficients <- coef(elastic_net_model, s = best_lambda)
#
# # Convert coefficients to a data frame
# coef_table <- data.frame(
#   Variable = rownames(coefficients),
#   Coefficient = as.numeric(coefficients)
# )
#
# # Filter for non-zero coefficients
# coef_table <- coef_table %>%
#   filter(Coefficient != 0) %>%
#   arrange(desc(abs(Coefficient)))
#
# # Print coefficients
# print(coef_table)
# Random Forest

pred_dta_x = pred_dta[,c("Hours_Studied", "Attendance", "Sleep_Hours",
```

```r
                              "Previous_Scores", "Tutoring_Sessions")] %>%
  as.data.frame
pred_dta_y = pred_dta$Exam_Score

# Hyperparameter tuning for number of variables randomly sampled
# as candidates at each split
set.seed(1)
tune_rf = tuneRF(x = pred_dta_x,
                 y = pred_dta_y,
                 ntreeTry=500)
m = tune_rf[which.min(tune_rf[,2]), 1]
m

# Cobstruct Random Forest
set.seed(123)
rf_m = randomForest(Exam_Score ~., data=pred_dta,
                    mtry=m, ntree = 1000,
                    keep.forest=TRUE, importance=TRUE)
# Mean prediction for each observation
pred_rf = predict(rf_m, type="response")
# OOB MSE from RF
plot(rf_m$mse)
rf_m$mse[1000]
# importance measure from RF
varImpPlot(rf_m, main="Importance plot of Random Forest")
# Single tree for RF
dat_rf_singletree = pred_dta %>% dplyr::select(-Exam_Score)
dat_rf_singletree$pred_rf = pred_rf

set.seed(1)
tree0_for_rf = rpart(pred_rf~., data=dat_rf_singletree, method="anova",
                     control=rpart.control(minsplit=5, cp=0.008))
#The two parameters in rpart.control() can be adjusted
#plotcp(tree0_for_rf)
#printcp(tree0_for_rf)
pred_single_tree_rf = predict(tree0_for_rf, newdata=dat_rf_singletree)
cor(pred_rf, pred_single_tree_rf) #0.898
# Visualize the single regression tree
rpart.plot(tree0_for_rf, fallen.leaves = FALSE, tweak=1.1, extra=101, under=TRUE)
# Gradient Boosting Machines (GBM) - Gradient Boosting Algorithm

# Load necessary libraries
```

```r
library(lightgbm)
library(Matrix)
library(dplyr)

# Preprocessing the data (similar to Elastic Net)
data <- na.omit(educdata)

# Specify target and predictors
target <- "Exam_Score"
categorical_cols <- names(data)[sapply(data, is.factor)]
numeric_cols <- names(data)[sapply(data, is.numeric) & names(data) != target]

# Create dummy variables for categorical predictors
dummies <- model.matrix(~ . - 1, data = data[categorical_cols])
X <- cbind(data[, numeric_cols], dummies)
y <- data[[target]]

# Split data into training and testing sets
set.seed(581)
train_index <- sample(1:nrow(X), 0.7 * nrow(X))
X_train <- X[train_index, ]
X_test <- X[-train_index, ]
y_train <- y[train_index]
y_test <- y[-train_index]

# Convert training and test sets to LightGBM format
lgb_train <- lgb.Dataset(data = as.matrix(X_train), label = y_train)
lgb_test <- lgb.Dataset(data = as.matrix(X_test), label = y_test, reference = lgb_train)

# Set parameters for the LightGBM model
params <- list(
  objective = "regression",
  metric = "rmse",
  boosting = "gbdt",           # Gradient Boosting Decision Trees
  learning_rate = 0.05,
  num_leaves = 31,
  max_depth = -1,
  feature_fraction = 0.9
)

# Train the LightGBM model
set.seed(581)
```

```
gbm_model <- lgb.train(
  params = params,
  data = lgb_train,
  nrounds = 1000,
  valids = list(train = lgb_train, test = lgb_test),
  early_stopping_rounds = 50,
  verbose = 1
)

# Predict on test data
predicted_values <- predict(gbm_model, as.matrix(X_test), num_iteration = gbm_model$best_ite

# Compute R-squared
rss <- sum((y_test - predicted_values)^2)  # Residual sum of squares
tss <- sum((y_test - mean(y_test))^2)      # Total sum of squares
r_squared <- 1 - rss / tss

# Compute RMSE
rmse <- sqrt(mean((y_test - predicted_values)^2))

# Print performance metrics
cat("R-squared on test data:", round(r_squared, 4), "\n")
cat("Root Mean Squared Error (RMSE):", round(rmse, 4), "\n")
library(kableExtra)
# Feature Importance
importance <- lgb.importance(gbm_model, percentage = TRUE)
importance[1:10,] %>% kable()

# Plot feature importance
lgb.plot.importance(importance, top_n = 12, measure = "Gain")
```