Predicting Academic Outcomes and Identifying Contributing Factors

COM SCI M148, Fall 2024

Maya Josifovska, Quinn Hilger, Kyan Kornfeld, John Reinker

I. Overview

This repository contains an analysis into the Student Performance dataset, a publicly available dataset containing various aspects of a student's life. These include numerical and categorical features such as:

- Hours Studied: time dedicated to preparation for the exam
- Attendance: percentage of classes attended
- Family Income: family income level
- Gender: gender of the student

The observational unit is a single student's record. There are 6606 observations and 19 features. The target variable is the exam score, which measures academic performance.

II. Problem Overview

The goal of this project is to develop a model that accurately predicts a student's exam score based on the provided features and identify the most significant factors contributing to the variability in the exam scores, which can be used to improve student outcomes.

III. Key Methodology

The key methodology of our project involved the following steps:

Preprocessing:
There were few NaN values in the dataset, meaning we opted to drop them instead of using imputation. Additionally, we used ordinal encoding to convert categorical variables into numeric values. This conversion method was chosen because all the categorical variables had some inherent ordering, which was preserved during the transformation. We opted for an 80% training, 10% validation, and 10% test split, as early tests of our model did not display any signs of overfitting.

Linear Regression:
To accomplish our primary goal of creating a model to predict a student's exam score, we trained a least squares linear regression model using the training set. To narrow down the number of features in the model, we performed various correlation testing and visual plotting to determine which features would be most useful in predicting exam scores. This concluded in the selection of the following 8 features based on magnitude of correlation and domain relevance.

- Hours Studied
- Attendance
- Parental Involvement
- Access to Resources
- Previous Scores
- Learning Disability
- Distance from Home
- Tutoring Sessions

Linear Regression was the final model used to predict exam score for various reasons. First, EDA revealed linear relationships between the above features and exam score, removing the requirement to use a higher order linear regression formula or more complex models such as a neural network. Second, interpretability of feature importance on a linear regression model provides clear insight as to how one could improve their exam score. Coefficients of our model on raw data (unscaled) could be used to quantify the impact of increased effort (in attendance and hours studied).

Feature Importance:
To accomplish our secondary goal of identifying the factors that contribute most to a student's exam scores, finding the correlation coefficient for each feature with respect to exam scores was our initial screen to reduce the number of features to 8. After the model was trained, because our input data was standardized, we used the coefficients of the features as feature importance measures.

Model Evaluation:
Our key model evaluation metrics were the Mean Squared Error (MSE), Root Mean Squared Error (rMSE), the R-squared value, the Correlation Coefficient and the Mean Absolute Error (MAE).

IV. Results

Following the training of the model, we computed our evaluation metrics on the validation set, followed by k-fold cross validation testing using 5 folds. The metrics returned displayed strong performance, with little difference between cross validation and validation set. These results can be seen in the table below.

Metric	**Validation Set**	**5-Fold Cross Validation**
MSE	4.9694	5.2979
MAD	0.9270	0.9642
R^2	0.6798	0.6578

Following metric evaluation of the validation set and cross validation, we decided on our final iteration of the linear regression model. To evaluate the final model's performance we calculate the same metrics, alongside correlation on the test set.
Overall, the evaluation metrics indicated a strong model for predicting exam scores. First, the R-squared coefficient was 0.754, indicating that the model explained a high proportion of the variability in the exam scores. Furthermore, the correlation coefficient was 0.870, showing a strong positive correlation and a close alignment with the actual exam scores. Additionally, the Mean Absolute Error was 0.883, indicating that the model's prediction was, on average, less than a point away from the actual score, bolstered by a small Root Mean Squared Error 0f 1.74. The coefficients of the model indicated the order of the importance of each feature was the following:

1. Attendance (2.285)
2. Hours Studied (1.759)
3. Parental Involvement (0.697)
4. Previous Scores (0.695)
5. Access to Resources (0.683)
6. Tutoring Sessions (0.607)
7. Distance from Home (-0.331)
8. Learning Disabilities (-0.234)

We can conclude, based on the above evaluation, that our linear regression model can accurately predict exam scores with a relatively small margin of error within the student performance dataset analyzed. This performance and clarity of feature importance proved more valuable to answering our problem statement than other data science methodologies applied to our data, leading us to select Linear Regression as our final model. The

simplicity of linear regression does carry some limitations. Our L2 loss function causes outliers to pull coefficients away from their true impact on exam score. The plot depicting predicted and actual scores shows some samples where students out-performed their expected exam score. These kinds of outliers are expected within the context of exam scores however. Future explorations of data can involve removing outliers or using an alternative loss function like L1 to reduce their impact.

V. How to Run the Code

The following resources are required, and can be accessed in the github repo available below the appendix in this document.

- StudentPerformanceFactors.csv
- cs148_final_project_code.ipynb

Steps to run code

1) Clone repository to gain access to the above files
2) Open cs148_final_project_code.ipynb in some code editor or using google colab (recommended). If using google colab, import the StudentPerformanceFactors.csv file into the local environment. Otherwise make sure the dataset csv is in the local directory opened in your code editor.
3) Running the code, in order to prepare data, train model, and evaluate performance, run each code block in order from top to bottom. Explanations of each code block in order are described below to assist users. Code blocks in the python notebook can be run by pressing the play button in the top left corner.
    a) Import required libraries for model training, evaluating and visualization
    b) Data Preprocessing: load csv into dataframe then clean, encode categorical variables, select important features, and construct train, validation, and test sets.
    c) Train linear regression model on L2 loss and print metrics when testing on the validation set
    d) Perform cross validation, printing metrics
    e) Evaluate metrics on test dataset
    f) Visualize predicted vs actual exam scores for the test set
    g) Conclusion: run the standardized model to print coefficients used to understand feature importance.

Appendix

Note: Code for the following analysis and data exploration is in each respective check-in notebook. However each notebook is updated with additional code and explanation beyond those submitted for check-ins. To see code for the appendix, please visit the notebook in italics for each topic.

I. Exploratory Domain Analysis *cs148_project.ipynb*

Prior to any model training, the first step involved exploring the StudentPerformanceFactors dataset. This involved sampling the complete raw data to observe what kind of features and corresponding values were present. From these observations we selected certain features to plot individually against the response variable of exam score. For continuous numerical features we created a scatter plot between the feature and exam score. The plot would also include a line of best fit. For each of these plots, we would look to see some linear relationship where the magnitude of the slope of the line of best fit was relatively large. These initial screenings gave us insight into the importance of features such as hours studied and attendance.

For categorical variables, we would create various box plots side by side for each possible value. In these plots we look for unique distributions amongst the varying categories. Especially for ordered categories, we looked for an increase or decrease in mean and quartiles as the categories increased in value. This allowed us to see relatively strong relationships between variables such as distance from home and exam score. On the other hand box plots such as sleep hours appeared to have little to no relationship with exam score.

Following these initial two variable plots, we plotted the correlation each feature has with exam score as a box plot. By adding various thresholds where correlation drops we were able to select 6 variables that had some correlation with exam score. Additionally, there were two features with negative correlation that had a decent magnitude. Although this correlation magnitude was less than some features we left out like parental education level, they described aspects of students' backgrounds not represented in the other features, therefore we chose to include these.

Following the exploratory data analysis, we divided up the raw dataset into 3 smaller datasets. Each split was generated using the train_test_split function from sklearn, in which it randomly samples the data to divide the dataframe in 2. The test split maintained 10% of the original data, and will be used after all training and evaluation of models, to provide an unbiased performance measure of the model. The validation set was also 10% of the original data, used to observe performance metrics during model training and tuning. Finally, the other 80% of the data is training data, which represents the samples the linear regression model is actually trained on. The final split of 80:10:10 was chosen after using various splits because there was little to no

evidence of overfitting. This allows us to include a larger training set in an attempt to capture all trends in the data.

II. Pre-Processing & Feature Engineering *cs148_project.ipynb*

To prepare our data for training, we performed various steps of data pre-processing. First we narrowed our data down to the 9 total features that will be used in our model. We did this to prevent removal of extra rows of data due to missingness in features we aren't using. By checking missingness for our main features we noticed 67 values missing for distance from home. Since these observations make up around ~1% of our dataset, we decided to drop these rows. Next we checked for any numeric columns that had negative values which aren't supported by their feature domain. This search resulted in no invalid values, meaning the rest of our dataset was clean.

To support our linear regression model and other methodologies such as clustering, we used ordinal encoding to convert categorical variables into numeric values. This conversion method was chosen because all the categorical variables had some inherent ordering, which was preserved during the transformation. This involved mapping categories such as "Near", "Moderate", and "Far" into 0,1,2. Other binary variables such as learning disabilities were converted from "Yes" and "No" to 1 and 0.

Normalization was not applied to the overall dataset, as we wanted to maintain all raw values. However throughout our analysis we would use the standard scalar to standardize the data when using coefficients to determine feature importance.

After performing these transformations and processing, we were left with a clean dataset that could be duplicated and used in all subsequent data analysis.

III. Linear Regression *cs148_project_checkin_2.ipynb*

In this project, we used least-squares linear regression for both the project check-in and our final code. In the check-in, we demonstrated linear regression with 2 variables, using the features we found had the strongest correlation with Exam Score. These linear regression models didn't perform great, as they were underfit due to the lack of features. However, since we were able to get decent performance out of a very simple model (correlation coefficient of around 0.6), this implied that linear regression could fit this data set well.

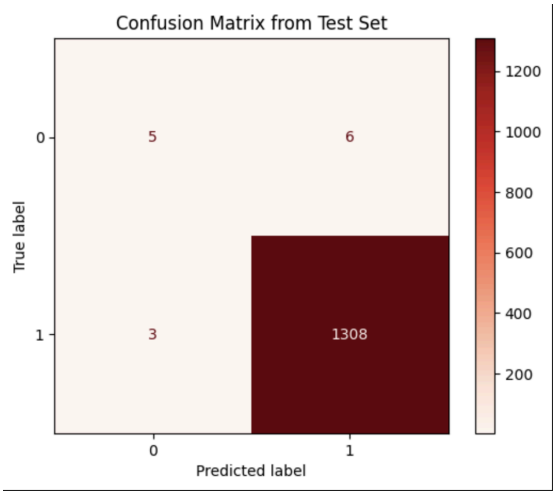
Once we decided that linear regression was the best method for our final project, we ran a similar model but using all 8 of our main features and were much more successful. To analyze feature importance, we normalized the data and sorted the coefficients of our linear regression model. The coefficients with the highest magnitude were Attendance and Hours Studied, which confirms our statistical and qualitative findings from EDA.

In both iterations of linear regression, regularization was not needed because we did not have a problem of overfitting. When we tried using Lasso or Ridge regression, all metrics, such as mean squared error, only got worse when compared to linear regression.

IV. Logistic Regression *cs148_project_checkin_3.ipynb*

Following the methodology used in the Linear Regression instance, Logistic Regression was performed on this dataset by first selecting 7 of the 8 most prominent predictor variables (I chose not to include "Distance_from_Home" since it was shown to include missing values in the data cleaning performed for Linear Regression). This is in contrast to the original Logistic Regression work done on the dataset for Check-In #3, where we used "Learning_Disability" as the sole predictor variable. Again, "Exam_Score" is the target variable; since Logistic Regression requires the dependent variable to be non-continuous and binary in nature, the "Exam_Score" column was mapped to either 0 or 1 using a value threshold of <60 (commonly used as a failing range). For scores 0-59, the mapping resulted in 0. Three of the predictor variables, which were categorical, were also one-hot encoded to numeric values.

Splitting and fitting was then performed, and the results of the Logistic Regression model were visualized using a confusion matrix, which can be seen below:



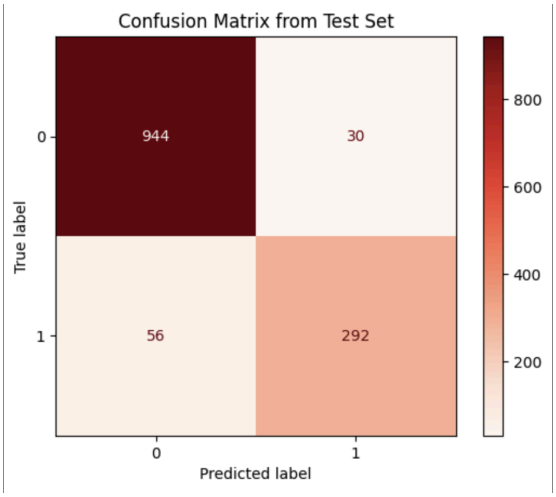
It is quite obvious that the model is essentially always predicting the "True" class label, but due to data class imbalance in the dataset this prediction actually works in the model's favor. The metrics generated are as follows:

Accuracy: 0.993
Prediction Error: 0.007
True Positive Rate (TPR): 0.998
True Negative Rate (TNR): 0.455

The model's overall accuracy is 99.3%, which is quite high and certainly laudable. However, the dataset's imbalance is highlighted by the fact that the True Negative Rate is less than 50%,

showing that, although the model is quite weak at correctly classifying "failing" students, it has little overall impact on accuracy due to there being such a small amount of failing students.

I bumped up the classification threshold to 70 to see how the model would respond. The new confusion matrix is as follows:



With the new metrics being:

Accuracy: 0.935
Prediction Error: 0.065
True Positive Rate (TPR): 0.839
True Negative Rate (TNR): 0.969

This new threshold has decreased the model's overall accuracy by 5.8% (95.3% → 93.5%). However, the balance between TPR and TNR is much better, with the TPR reducing by 15.9% (99.8% → 83.9%) in order to increase the TNR by a whopping 51.4% (45.5% → 96.9%).

The data was standardized, yet the act of examining feature importance in this case is difficult due to the issues posed by the class imbalances. However, one major observation regarding the features comes from comparing the results of the Logistic Regression applied here with the results of the Logistic Regression applied using only "Learning_Disability" as a predictor variable. See the metrics for the original application here:

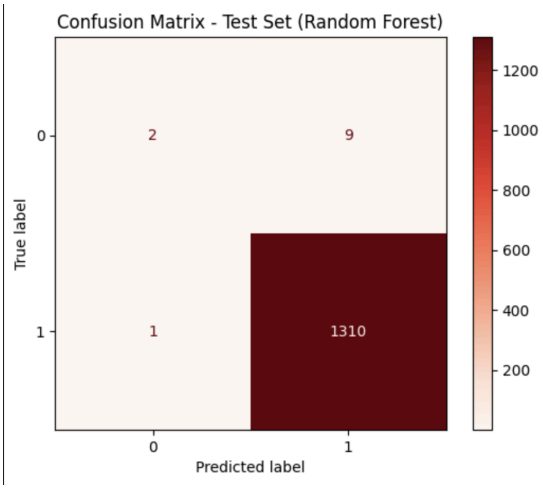
Accuracy: 0.890
Prediction Error: 0.110
True Positive Rate (TPR): 0.00
True Negative Rate (TNR): 1.00

The main takeaway is that "Learning_Disability" on its own was not the best feature for predicting the "Exam_Score"; the combination of the 7 most correlated features – which "Learning_Disability" is a part of – resulted in an overall accuracy increase of 10.3% (89.0% → 99.3%), which is significant. We also see that the model adjusted to not only predict the Negative class label, as in the old application it made this prediction every single time.

The model was fit both with and without L2 Regularization, but it had no effect; this is most likely due to being overshadowed by class imbalance issues in the dataset.

V. KNN & Decision Trees *cs148_project_checkin_4.ipynb*

The same changes applied for Logistic Regression were applied for training a Random Forest classifier model. This included expanding the set of predictor variables from "Learning_Disability" to the set of the 7 variables most correlated with "Exam_Score". Again, a threshold of 60 for classifying an "Exam_Score" as either 0 or 1 was used to initially train the model, with testing results being plotted as a confusion matrix:

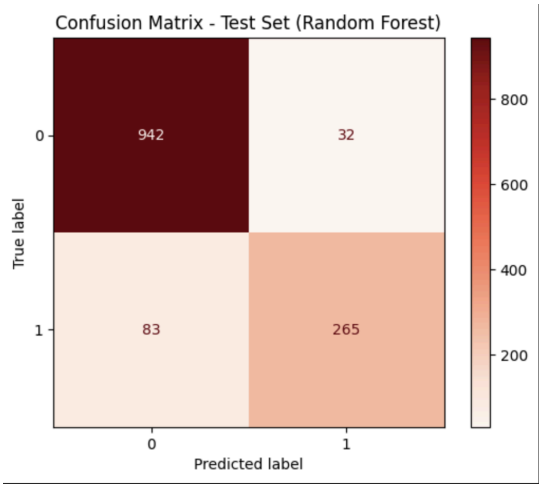


And the metrics were:

Accuracy: 0.990
Prediction Error: 0.010
True Positive Rate (TPR): 1.00
True Negative Rate (TNR): 0.180

The numbers are telling a similar story here when compared to Logistic Regression. The model is overall very accurate at 99%, but the True Negative Rate is very low at only 18%. But, yet again, this gets buried by the class imbalances in the dataset, so predicting the True class label results in high performance.

Once again, I changed our "hyperparameter" here – the classification threshold – from 60 to 70 and retrained the model to get the following confusion matrix:



With these metrics:

Accuracy: 0.910
Prediction Error: 0.090
True Positive Rate (TPR): 0.760
True Negative Rate (TNR): 0.970

Once more, we see this change sacrifice overall accuracy (a decrease of 9%) in exchange for a far better balance between the True Positive Rate and True Negative Rate, with the model now mostly predicting the Negative label.

5-Fold Cross Validation was performed for both instances of the hyperparameter, and the average accuracy across them was 99% and 90% respectively, which aligns with the metrics reported along with the confusion matrices above. No explicit Regularization techniques were applied across the training instances, and it most likely would not have much effect anyway – similar to in Logistic Regression.

Comparing the new results with the old results obtained by just using "Learning_Disability" as the predictor let's us gain more insight to the relationships between the features at play in this dataset:

Accuracy: 0.890
Prediction Error: 0.110
True Positive Rate (TPR): 0.009
True Negative Rate (TNR): 0.999
Average Accuracy across folds: 0.887

We can see that expanding the predictors has increased all our metrics across the board (just like it had done in Logistic Regression); the model's overall accuracy has increased 10%, which reiterates that "Learning_Disability," although being one of the variables most correlated with Exam_Score, is not solely sufficient for training a classification model on this dataset.

VI. Clustering & PCA *cs148_project_checkin_5.ipynb*

Both clustering and PCA were applied to the dataset to learn more about the data, rather than assist in our main target of exam score prediction. Clustering was performed with both

k-means clustering and agglomerative clustering. Both of these methods were analyzed using 2 clusters. The cluster count was decided by plotting the number of clusters against within cluster sum of squares, and analyzing where the elbow occurs. The elbow occurs at 2 clusters, therefore leading to the decision to use k=2. After clustering we printed out the average of each feature in the cluster, making up the centroid. This depicts a clear narrative surrounding exam scores. Cluster 0 attends class much more (88%) and studies much longer (21.9 hours) on average, resulting in a mean exam score of 70.2. On the other hand cluster 1 only attends class 72.4% of the time and studies 18 hours on average, resulting in a lower exam score of 64.6. There's also a distinguished difference related to family income, parental education level, etc. that differentiate these two clusters. We then plotted each cluster in a scatter plot along the attendance and hours studied graphs to see each group. The clustering used can assist in describing the narrative learned by feature importance in linear regression, where an increase in studying and class attendance can place you in the cohort that scores higher on the exam.

PCA was also applied to simply learn more about feature importance. First we standardized all features then applied PCA to just observe the first 4 principal components. By printing out the variable loadings, we see the similar narrative of hours studied and attendance having relatively large coefficients. Additionally the exam score coefficients appear quite large, proving it can describe a large amount of the variance in the data. This helps us believe that exam scores have trends to be discovered within our dataset.

Overall, unsupervised learning techniques do not apply directly to our problem statement very well. However they provide interesting insight to back up our claims of feature importance. Additionally, they can provide interesting visual depictions, specifically clustering, that can be used to spread the conclusion of our data. That conclusion being studying longer and attending class more often.

VII. Neural Network *cs148_project_checkin_6.ipynb*

The neural network implemented in the project is a simple feedforward neural network designed to predict exam scores using all of the features in the dataset. In the preprocessing stage, rows with missing values were removed and the categorical features were transformed using one-hot encoding. The dataset was split on an 80-20 test split and then mean-scaled to zero.

The architecture of the model consists of an input layer, a hidden layer with 64 neurons and a single output node. In the hidden layer, the reLU activation function was applied to introduce non-linearity. The Mean Squared Error (MSE) function was used as a loss metric, and the optimizer used to update the model's weights during training was Stochastic Gradient Descent (SGD).

Initial experiments were completed with a learning rate of 0.01 and 100 epochs. However, analysis of the training process using trace plots indicated that the learning rate of 0.01 caused the loss to oscillate, causing instability. With a learning rate of 0.001, the loss decreased

towards zero steadily, and then remained stable. Thus, the learning rate of 0.001 was selected. Additionally, 100 epochs was sufficient to observe this pattern, and increasing the epoch count did not significantly improve model performance.

Validation loss appeared to follow the training loss, indicating the model was not overfitting. The R^2 of the model was 0.4393, indicating that it explained a reasonable proportion of the model's effectiveness, though also indicating room for improvement, possibly through the use of a more complex model architecture.

VIII. Hyperparameter Tuning

When attempting to use regularization in linear regression, we needed to tune the penalty hyperparameter, alpha. Using the default value of 0.1 achieved nothing, so we tried other values of various magnitudes, to no effect. Finally, we graphed the performance (MSE) of models trained with different hyperparameters and found the minimum MSE was with the lowest possible alpha value. This proves that our linear regression model does not benefit from regularization. *(cs148_project_checkin_2.ipynb)*

For the neural network hyperparameter tuning, a trace plot was used to evaluate the number of epochs and the learning rate. The number of epochs was initialized to 100 and the learning rate was .01. The initial learning rate of 0.01 caused the loss value to oscillate. With a smaller learning rate of 0.001, the loss steadily decreased and then stabilized, remaining relatively constant. 100 epochs appeared to be enough to observe this; additional epochs did not have a notable effect on the loss or the R^2. *(cs148_project_checkin_6.ipynb)*

Additional hyperparameter tuning was performed during clustering, to find the optimal number of clusters to describe the data. This was performed by using the elbow method to observe the k value that gains the biggest decrease in the within cluster sum of squares metric.

Resources

Github Link: https://github.com/QuinnHilger/cs148-student-performance
Main Final Notebook
cs148_project.ipynb
cs148_project_checkin_2.ipynb
cs148_project_checkin_3.ipynb
cs148_project_checkin_4.ipynb
cs148_project_checkin_5.ipynb
cs148_project_checkin_6.ipynb

**Please find all related python notebooks in the above github repo.*