A logo for college computing

Description automatically generated

**Assessment Cover Page**

|  |  |
| --- | --- |
| *Student Full Name* | Derek O’Hara |
| *Student Number* | sbs24018 |
| *Module Title* | Machine Learning |
| *Assessment Title* | CA1 Project |
| *Assessment Due Date* | 28th April 2024 |
| *Date of Submission* | 28th April 2024 |

**Declaration**

By submitting this assessment, I confirm that I have read the CCT policy on academic misconduct and understand the implications of submitting work that is not my own or does not appropriately reference material taken from a third party or other source.

I declare it to be my own work and that all material from third parties has been appropriately referenced.

I further confirm that this work has not previously been submitted for assessment by myself or someone else in CCT College Dublin or any other higher education institution.

Contents

[Introduction 1](#_Toc165215212)

[Data Characterization and Pre-processing 1](#_Toc165215213)

[Analysis and Machine Learning Models 12](#_Toc165215214)

[Linear Regression 14](#_Toc165215215)

[Observations: 15](#_Toc165215216)

[Support Vector Machine Regression 16](#_Toc165215217)

[Observations: 17](#_Toc165215218)

[Decision Tree Regressor 18](#_Toc165215219)

[Observations: 19](#_Toc165215220)

[Random Forest Regressor 20](#_Toc165215221)

[Observations: 21](#_Toc165215222)

[Model Comparison Summary 24](#_Toc165215223)

[Hyperparameter Tuning 25](#_Toc165215224)

[Model Comparison Summary after Hyperparameter Tuning 28](#_Toc165215225)

[Conclusion 29](#_Toc165215226)

[References 30](#_Toc165215227)

[Dataset: 30](#_Toc165215228)

[Github: 30](#_Toc165215229)

[Word Count 30](#_Toc165215230)

[Code comments: 31](#_Toc165215231)

# Introduction

Analysing and predicting student performance is crucial for educational institutions to improve efficiency. It helps identify students with low academic achievements early on, high dropout rates, and delays in graduation (Daniel, B. 2015). It is very important for educational institutions, to understand the potential of using collected data to improve the learning efficacy and academic achievements of both the individual student and institutions themselves (M. Nachouki and M. Abou Naaj, 2022).

This report aims to explore and demonstrate the application of machine learning algorithms to predict academic success. I will be focusing on predicting a group of students' final grade “**G3**” in a course based on their prior academic data and demographic features.

The dataset I have chosen is called the “Student Performance Dataset” and can be found at [kaggle.com](https://www.kaggle.com/datasets/devansodariya/student-performance-data). As I become more familiar with the data and using prediction and classification algorithms where appropriate, I hope to be able to identify patterns and insights from the data to achieve my goal. (word count = 160)

# Data Characterization and Pre-processing

This dataset was compiled through a survey of students' math course in secondary school. It is made up of 33 columns/features of both numerical and categorical types and 395 rows. The features comprise of several considerations such as the students past academic performance, demographics, and social factors.

Data preprocessing refers to the technique of preparing (cleaning and organizing) the raw data to make it suitable for a building and training Machine Learning models (K Goyal 2024). First steps are to load and display the main features of the dataset to get an initial understanding and identify any obvious relationships or anomalies.

A screenshot of a computer

Description automatically generated

Data types show encoding categorical variables will be necessary. No null values in the dataset means no need to implement any missing value handling techniques.

A screenshot of a computer

Description automatically generated A screenshot of a computer

Description automatically generated

Metrics of the numerical columns:

A screenshot of a computer

Description automatically generated

A Boxplot of the numerical columns helps identify any distint outliers in the data.

A screen shot of a graph

Description automatically generated

Variable with the widest range of values with a lot of outliers is **absences**, indicating that most students have few absences, but some have a lot more than typical.

**G1, G2 & G3** are spread out with several outliers, particularly for the final grade **G3**. This suggests varied performance among students, with some scoring much higher or lower than the typical range.

To deal with outliers effectively, I can:

* **Keep them**: If they represent valid variations.
* **Remove them**: If they're due to errors or if their extreme values could skew my analysis.
* **Transform them**: Apply a transformation to reduce the impact of outliers, for instance, using a log transformation (S. Anuganti, 2020).

I visualise the distribution of **G3** values by means of a histogram using matplotlib and seaborn libraries. A significant left skew of the **G3** grade distribution is caused by 38 zero entries.

A graph with blue lines and a blue line

Description automatically generated

I similarly plot the distribution of all the numerical features.

A screenshot of a graph

Description automatically generated

The correlation matrix heatmap shows that the strongest correlation between **G3** marks is with the marks obtained in **G1** & **G2**.

A screenshot of a computer

Description automatically generatedWe also see negative correlation between **failures** and **G1**, **G2**, **G3**, which indicates that students with more past failures tend to have lower current grades.

**studytime** negative correlated with **failures** suggests that more study time is associated with fewer failures.

I perform a thorough exploratory data analysis by creating visual relationships between each feature and the target variable, **G3**.

Scatter plots visualise numerical features. A screenshot of a computer screen

Description automatically generatedA group of graphs with numbers

Description automatically generated with medium confidence

A group of graphs with dots

Description automatically generated with medium confidence

Bar charts to plot the mean of **G3** for each categorical feature.

A group of colored bars

Description automatically generated with medium confidence

A group of colored bars

Description automatically generated

A group of colored bars

Description automatically generated with medium confidence

I picked some of the main features and created pair plots to further analyse patterns. The strongest relationships to **G3** are again with **G1** & **G2**.

A screenshot of a computer screen

Description automatically generated

Taking **studytime**, we can see increasing trend in median grades as study time increases but the overall spread is similar. We also see the outliers across all levels were study time did not influence a zero mark.

A screenshot of a graph

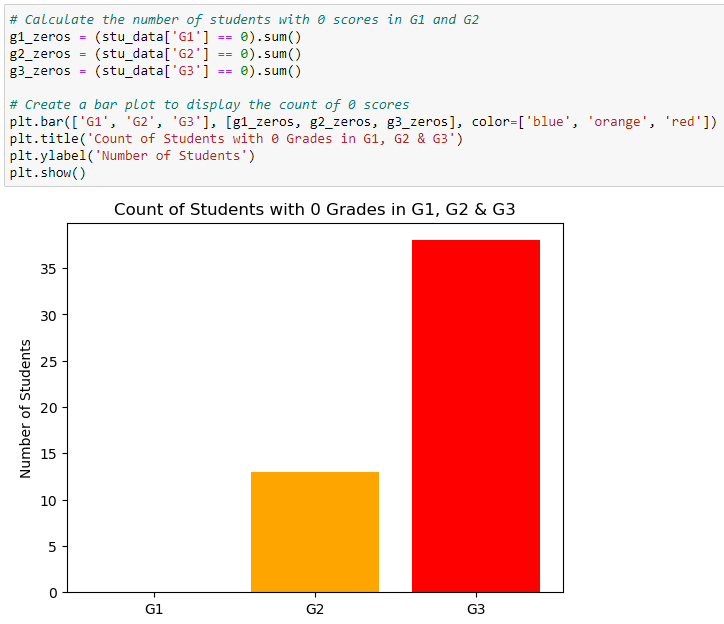
Description automatically generated

I encoded the categorical variables using one hot encoding.

A screenshot of a computer

Description automatically generated

Between **G1**, **G2** & **G3,** the number of zero grades have increased to 38.



Based on my analysis, I assume that the 38 students did not sit the exam.

A graph with blue lines and numbers

Description automatically generated

After dropping the 38 zero **G3** grades, we can see that the distribution skew has improved.

(word count = 568)

# Analysis and Machine Learning Models

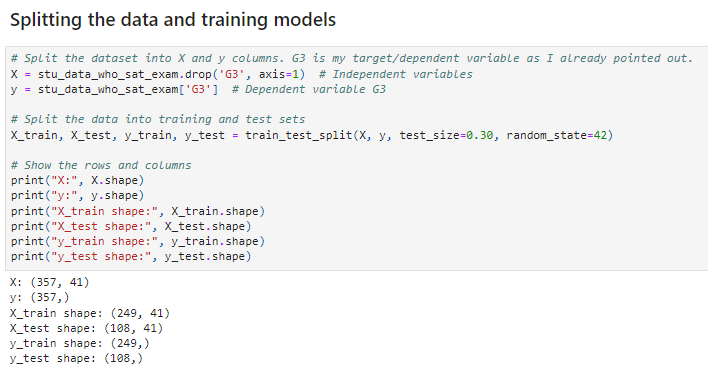
I tested four regression models to predict the final grade **G3**.

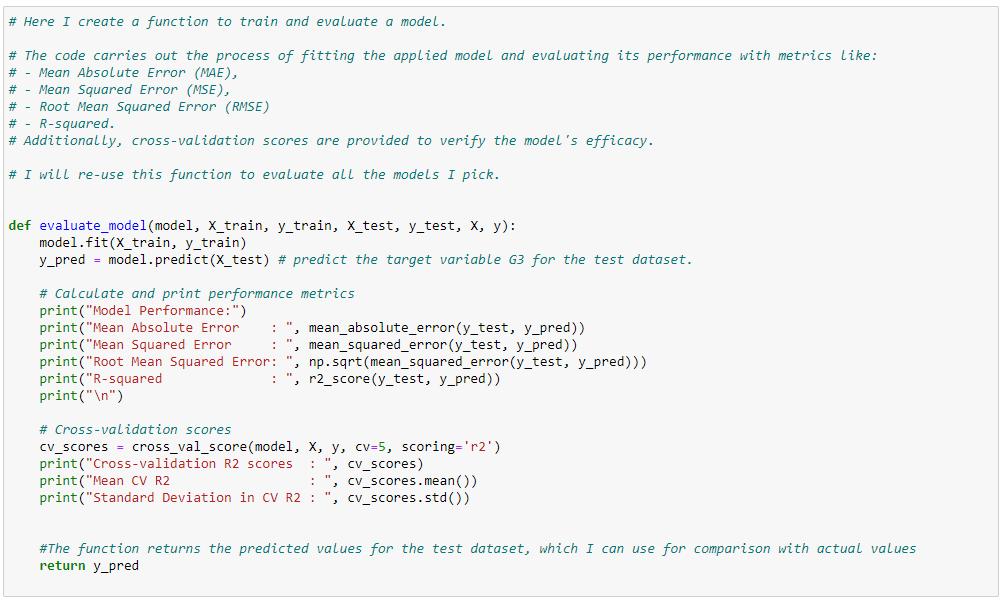
Models used:

* Linear Regression.
* Support Vector Machine Regression.
* Decision Tree Regressor.
* Random Forest Regressor.

These models were chosen based on their ability to handle different types of data distributions and robustness in regression tasks.

Each model was trained using different splits of the data (20%, 25%, and 30%) to ensure robustness and reliability of the predictions across various training scenarios.



Cross-validation was used to validate the results, ensuring that the models were not overfitting and were generalisable to new data. 

Using ExtraTreesRegressor, I find the important features that influence the prediction of **G3**.

A screenshot of a computer

Description automatically generated

**G2** & **G1** grades are the strongest indicators of **G3**

## Linear Regression

A screenshot of a graph

Description automatically generated

### Observations:

* R-squared above 0.90 across all splits, indicating a good fit.
* Small standard deviation in cross-validation scores suggests model performance is stable across different subsets of the data.
* Relatively low mean absolute and squared errors, indicating good accuracy.
* The variance of the residuals is consistent across the range of predictions, which is a good sign of model fit.

A screenshot of a computer

Description automatically generated

## Support Vector Machine Regression

A screenshot of a graph

Description automatically generated

### Observations:

* Similar R-squared values to Linear Regression, indicating a good fit.
* Cross-validation results are consistent across different splits, showing good generalization.
* Slightly higher errors compared to Linear Regression but still performs well.
* Predictions closely aligned with the actual grades, although the fit is not as perfect as the dashed line, indicating room for improvement.
* Residuals indicate a good fit, although the spread seem to increase slightly with the predicted grade.

A screenshot of a computer

Description automatically generated

## Decision Tree Regressor

A screenshot of a graph

Description automatically generated

### Observations:

* R-squared notably lower compared to other models, suggesting it does not capture the variance in the target as well.
* Mean CV R2 significantly lower than other models, and the standard deviation is higher, which indicates overfitting and less stability.
* Higher errors than Linear Regression and SVM, indicating less accuracy.

A screenshot of a computer

Description automatically generated

## Random Forest Regressor

A screenshot of a graph

Description automatically generated

### Observations:

* Exhibits the lowest errors among all models, suggesting it's the most accurate.
* Highest R-squared values, especially with the 30% split, indicating a very good fit.
* Cross-validation scores are very high but show a little more variability compared to Linear Regression, yet this model still appears to generalise well.
* Strong positive correlation between predicted and actual values, indicating good model performance.
* Residual Plot suggests some patterns in residuals that may need further investigation to improve model predictions.

A screenshot of a computer

Description automatically generated

The validation curve also provides valuable insight into the performance of the Random Forest model as we vary the number of estimators (trees in the forest).

A screenshot of a computer

Description automatically generated

Overfitting would be indicated by a large gap between the training and validation scores. However, the gap here is relatively small, especially in the range beyond 20 trees. This suggests that the model is not severely overfitting.

Underfitting would be indicated by both scores being low. That's not the case here as both scores are quite high.

Importance features on the trained Random Forest model show **G2** is by far the most prominent.

A screen shot of a computer code

Description automatically generated

## Model Comparison Summary

The Random Forest model, particularly with a 30% split, seems to be the best performer in terms of both error metrics and R-squared. The model has not only fit the training data well but also shown good performance on cross-validation, indicating that it should generalize well to unseen data.

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Model** | **Metric** | **Split 20%** | **Split 25%** | **Split 30%** |
| **Linear Regression** | Mean Absolute Error | 0.67482145 | 0.719230638 | 0.720122526 |
| Mean Squared Error | 0.80939602 | 0.883210251 | 0.884019061 |
| Root Mean Squared Error | 0.899664393 | 0.939792664 | 0.940222878 |
| R-squared | 0.915986045 | 0.908474451 | 0.912569543 |
| Cross-validation R2 scores | [0.92048806 0.9338324 0.91119051 0.93778319 0.91819327] | [0.92048806 0.9338324 0.91119051 0.93778319 0.91819327] | [0.92048806 0.9338324 0.91119051 0.93778319 0.91819327] |
| Mean CV R2 | 0.924297487 | 0.924297487 | 0.924297487 |
| Standard Deviation in CV R2 | 0.009963381 | 0.009963381 | 0.009963381 |
| **SVM** | Mean Absolute Error | 0.678925458 | 0.700780294 | 0.742567348 |
| Mean Squared Error | 0.795795817 | 0.838678055 | 0.919241518 |
| Root Mean Squared Error | 0.892073885 | 0.915793675 | 0.958770837 |
| R-squared | 0.917397723 | 0.913089245 | 0.909086004 |
| Cross-validation R2 scores | [0.91920965 0.93456898 0.91065012 0.92628461 0.91095953] | [0.91920965 0.93456898 0.91065012 0.92628461 0.91095953] | [0.91920965 0.93456898 0.91065012 0.92628461 0.91095953] |
| Mean CV R2 | 0.920334576 | 0.920334576 | 0.920334576 |
| Standard Deviation in CV R2 | 0.009175694 | 0.009175694 | 0.009175694 |
| **Decision Tree** | Mean Absolute Error | 0.736111111 | 0.811111111 | 0.787037037 |
| Mean Squared Error | 1.013888889 | 1.255555556 | 1.083333333 |
| Root Mean Squared Error | 1.006920498 | 1.120515754 | 1.040833 |
| R-squared | 0.894760026 | 0.869888951 | 0.892857143 |
| Cross-validation R2 scores | [0.86270158 0.88920617 0.8318575 0.84002832 0.85215178] | [0.86270158 0.88920617 0.8318575 0.84002832 0.85215178] | [0.86270158 0.88920617 0.8318575 0.84002832 0.85215178] |
| Mean CV R2 | 0.855189071 | 0.855189071 | 0.855189071 |
| Standard Deviation in CV R2 | 0.019985177 | 0.019985177 | 0.019985177 |
| **Random Forest** | Mean Absolute Error | 0.611111111 | 0.638888889 | 0.630185185 |
| Mean Squared Error | 0.688830556 | 0.753157778 | 0.709588889 |
| Root Mean Squared Error | 0.829958165 | 0.867846633 | 0.842370992 |
| R-squared | 0.928500539 | 0.921951563 | 0.929820879 |
| Cross-validation R2 scores | [0.91232561 0.94254654 0.89575837 0.93189747 0.91709329] | [0.91339417 0.94391342 0.90000931 0.93431761 0.91579748] | [0.9137083 0.9413076 0.89519166 0.93093855 0.91358797] |
| Mean CV R2 | 0.919924257 | 0.921486399 | 0.918946816 |
| Standard Deviation in CV R2 | 0.016154686 | 0.01566349 | 0.015901402 |

# Hyperparameter Tuning

Hyperparameter tuning is the process of determining the right combination of hyperparameters that maximizes the model performance (Shahul ES and A. Bajaj 2023).

It works by:

* Run multiple trials in a single training process.
* Each trial tests the chosen hyperparameters with values within the limits specified.
* The result is a set of hyperparameter values that are best suited for the model to give optimal results.

I used GridSearchCV to perform hyperparameter tuning for my Random Forest Regressor model.

My first attempt actually resulted in model performance decreasing across most metrics. A screenshot of a computer code

Description automatically generated

Second attempt, after increasing my parameters, I got a slightly improved model than the original.

A screenshot of a computer program

Description automatically generated

A screenshot of a graph

Description automatically generated

## Model Comparison Summary after Hyperparameter Tuning

The second set of Random Forest tuned parameters performs better than the first but is comparable to the untuned model, suggesting that the initial parameters might be close to optimal.

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| **Model** | **Metric** | **Split 20%** | **Split 25%** | **Split 30%** | **Hyperparameter Tuning 1 - 30%** | **Hyperparameter Tuning 2 - 30%** | |
| **Linear Regression** | Mean Absolute Error | 0.67482145 | 0.719230638 | 0.720122526 |  | | |
| Mean Squared Error | 0.80939602 | 0.883210251 | 0.884019061 |
| Root Mean Squared Error | 0.899664393 | 0.939792664 | 0.940222878 |
| R-squared | 0.915986045 | 0.908474451 | 0.912569543 |
| Cross-validation R2 scores | [0.92048806 0.9338324 0.91119051 0.93778319 0.91819327] | [0.92048806 0.9338324 0.91119051 0.93778319 0.91819327] | [0.92048806 0.9338324 0.91119051 0.93778319 0.91819327] |  | | |
| Mean CV R2 | 0.924297487 | 0.924297487 | 0.924297487 |
| Standard Deviation in CV R2 | 0.009963381 | 0.009963381 | 0.009963381 |
| **SVM** | Mean Absolute Error | 0.678925458 | 0.700780294 | 0.742567348 |  | | |
| Mean Squared Error | 0.795795817 | 0.838678055 | 0.919241518 |
| Root Mean Squared Error | 0.892073885 | 0.915793675 | 0.958770837 |
| R-squared | 0.917397723 | 0.913089245 | 0.909086004 |
| Cross-validation R2 scores | [0.91920965 0.93456898 0.91065012 0.92628461 0.91095953] | [0.91920965 0.93456898 0.91065012 0.92628461 0.91095953] | [0.91920965 0.93456898 0.91065012 0.92628461 0.91095953] |  | | |
| Mean CV R2 | 0.920334576 | 0.920334576 | 0.920334576 |
| Standard Deviation in CV R2 | 0.009175694 | 0.009175694 | 0.009175694 |
| **Decision Tree** | Mean Absolute Error | 0.736111111 | 0.811111111 | 0.787037037 |  | | |
| Mean Squared Error | 1.013888889 | 1.255555556 | 1.083333333 |
| Root Mean Squared Error | 1.006920498 | 1.120515754 | 1.040833 |
| R-squared | 0.894760026 | 0.869888951 | 0.892857143 |
| Cross-validation R2 scores | [0.86270158 0.88920617 0.8318575 0.84002832 0.85215178] | [0.86270158 0.88920617 0.8318575 0.84002832 0.85215178] | [0.86270158 0.88920617 0.8318575 0.84002832 0.85215178] |  | | |
| Mean CV R2 | 0.855189071 | 0.855189071 | 0.855189071 |
| Standard Deviation in CV R2 | 0.019985177 | 0.019985177 | 0.019985177 |
| **Random Forest** | Mean Absolute Error | 0.611111111 | 0.638888889 | 0.630185185 | 0.977037625 | | 0.62158630044 |
| Mean Squared Error | 0.688830556 | 0.753157778 | 0.709588889 | 1.6710302 | | 0.70437644169 |
| Root Mean Squared Error | 0.829958165 | 0.867846633 | 0.842370992 | 1.292683333 | | 0.83927137547 |
| R-squared | 0.928500539 | 0.921951563 | 0.929820879 | 0.834733277 | | 0.93033639587 |
| Cross-validation R2 scores | [0.91232561 0.94254654 0.89575837 0.93189747 0.91709329] | [0.91339417 0.94391342 0.90000931 0.93431761 0.91579748] | [0.9137083 0.9413076 0.89519166 0.93093855 0.91358797] | [0.80674425 0.81211974 0.80560448 0.81309717 0.83063157] | | [0.91148225 0.94027011 0.89887866 0.93087874 0.9238684 ] |
| Mean CV R2 | 0.919924257 | 0.921486399 | 0.918946816 | 0.813639442 | | 0.92107563237 |
| Standard Deviation in CV R2 | 0.016154686 | 0.01566349 | 0.015901402 | 0.008982651 | | 0.01454006349 |

(word count = 600)

# Conclusion

This project successfully demonstrates the application of machine learning techniques to predict educational outcomes. The Random Forest Regressor, with optimized hyperparameters, proved marginally to be the most effective model. This analysis not only provides a tool for predicting student performance but also offers insights that could help in educational planning and student support.

The project adheres to the ethical considerations of machine learning, ensuring that the predictive models do not discriminate based on sensitive demographic features by focusing solely on academic-related variables. Future work could explore the reason 38 students did not sit the final exam.

(word count = 96)

# References

Daniel, Ben. (2015). Big Data and analytics in higher education: Opportunities and challenges. British Journal of Educational Technology. 46. 10.1111/bjet.12230. Available at: <https://www.researchgate.net/publication/269936924_Big_Data_and_analytics_in_higher_education_Opportunities_and_challenges>

Nachouki, Mirna & abou naaj, Mahmoud. (2022). Predicting Student Performance to Improve Academic Advising Using the Random Forest Algorithm. International Journal of Distance Education Technologies. 20. 17. 10.4018/IJDET.296702. Available at: <https://www.researchgate.net/publication/362839521_Predicting_Student_Performance_to_Improve_Academic_Advising_Using_the_Random_Forest_Algorithm>

Kechit Goyal (2024) 'Data Preprocessing in Machine Learning'. Available at: [https://www.upgrad.com/blog/data-preprocessing-in-machine-learning](https://www.upgrad.com/blog/data-preprocessing-in-machine-learning/)

Suresh Anuganti (2020). How to remove outliers for machine learning. Available at: <https://medium.com/analytics-vidhya/how-to-remove-outliers-for-machine-learning-24620c4657e8>

Shahul ES and Aayush Bajaj (2023) 'Hyperparameter Tuning in Python: A Complete Guide', *Neptune.ai Blog*. Available at: <https://neptune.ai/blog/hyperparameter-tuning-in-python-complete-guide>

## Dataset:

Student Performance Dataset (2022) <https://www.kaggle.com/datasets/devansodariya/student-performance-data/data>

## Github:

<https://github.com/derekoharacct/Machine-Learning-CA-1.git>

## Word Count

Total word count = 1424

## Code comments:

# Suppress the warnings

# Here I import all the needed libraries, classes and modules that I use throughout the notebook.

# As i progress and need to import anything in the later code cells, I return here and add it to this cell so everything is loaded from the beginning.

# Load the dataset

# Display the first 10 rows of the dataset.

# Display dataset info.

# Describe dataset numerical column data.

# Here I create a Boxplot of the numerical columns in order to identify early if there are any outliers (Individual Dots) in the data.

# Checking null values for all columns.

# Identify the numeric and categorical columns in the dataset:

# Identify all categorical columns in the stu\_data

# Here I want to visualize the distribution of G3 final exam grades by histogram using matplotlib and seaborn libraries.

# G3 final grade is my focus for this project.

# I also want to create histograms to see the distribution of each numerical feature in the dataframe.

# Here I calculate and plot the correlation matrix as a heatmap in order to easily identify the correlation between two features

# In the code section, I try to perform a thorough exploratory data analysis by creating visual relationships between each feature and the target variable, G3.

# This will hopefully allow for an investigation into potential linear relationships or category-wise differences in the dataset.

# Identify numeric and categorical columns, excluding 'G3' from numeric if present

# Setting the plotting area and dimensions

# Plotting numeric features with scatter plots

# Plotting categorical features with bar charts

# Adjust layout and remove unused subplots if any

# Creating pair plots for a subset of the main features.

# Here I create Box plots to compare distributions of grades based on study time.

# Intuition and common sense would say that there should be a correlation between study time and grade performance.

# Maybe not in this case.

# Encoding Categorical variables into numerical variables using One hot encoding

# Display the first 10 rows of the dataset.

# Here I am comparing the numerical & categorical pre and post One hot encoding

# Print all columns after encoding

# Plotting boxplots for numerical features after encoding to identify any new outliers

# Calculate the number of students with 0 scores in G1 and G2

# Create a bar plot to display the count of 0 scores

# I am going to remove the 38 cases where G3 is zero because I am making the assumption that these students did not sit the exam

# I again visualize the distribution of G3 grades after I exclude the students who did not sit the exam

# Split the dataset into X and y columns. G3 is my target/dependent variable as I already pointed out.

# Split the data into training and test sets

# Show the rows and columns

# Get the important features that influence the prediction of students' final grades G3 using ExtraTreesRegressor

#print(sat\_exam\_selection.feature\_importances\_)

# Here I create a function to train and evaluate a model.

# The code carries out the process of fitting the applied model and evaluating its performance with metrics like:

# - Mean Absolute Error (MAE),

# - Mean Squared Error (MSE),

# - Root Mean Squared Error (RMSE)

# - R-squared.

# Additionally, cross-validation scores are provided to verify the model's efficacy.

# I will re-use this function to evaluate all the models I pick.

# predict the target variable G3 for the test dataset.

# Calculate and print performance metrics

# Cross-validation scores

#The function returns the predicted values for the test dataset, which I can use for comparison with actual values

# I also need a function to create a visual comparison between actual and predicted values from the model I have trained.

# I will use a scatter plot to show this

# This line adds a diagonal line to the plot, which represents the line of perfect prediction. If every predicted value was exactly equal to the actual value, all points would lie on this line.

# Take the actual and predicted values as inputs and display a residual plot.

# The residuals are calculated as the difference between the actual and the predicted values.

# Here I train & evaluate the Linear Regression model lr\_sat\_exam

# Then I call the function to plot Actual v Predicted to get the visual comparison between actual and predictedvalues

# Training & evaluating the SVM model

# Call the function to plot Actual v Predicted for svm\_model

# Call the function to plot the residuals between Actual v Predicted for svm\_model

# Training & evaluating the Decision Tree model

# Call the function to plot Actual v Predicted

# Call the function to plot the residuals between Actual v Predicted

# Training & evaluating the random forest model

# Call the function to plot Actual v Predicted

# Call the function to plot the residuals between Actual v Predicted

# In this block of code I determin the optimal number of trees (n\_estimators) in a RandomForestRegressor model by plotting a validation curve.

# Prepare the range of 'n\_estimators' to test

# Calculate the scores across the range of 'n\_estimators'

# Calculate mean and standard deviation for training set scores

# Calculate mean and standard deviation for test set scores

# Plot the validation curve

# Plot mean accuracy scores for training and test sets

# Plot the std deviation as a shaded area around the mean

# Create plot

# Here I want to extract and visualize feature importance from the trained Random Forest model.

# These importances are based on how much each feature decreases the impurity of the split (the Gini impurity).

# Here I use GridSearchCV to perform hyperparameter tuning for a RandomForestRegressor.

# It is a search over specified parameter values for an estimator, intended to find the combination of parameters that

# will return the best model performance.

# Set up the refined parameter grid based on initial findings (Hyperparameter Tuning 1)

#param\_grid = {

# 'n\_estimators': [150, 200, 250], # The number of trees in the forest

# 'max\_features': ['sqrt'], # Suppose 'sqrt' was the best in the initial search

# 'max\_depth': [20, 30, 40], # Narrowing down based on initial results

# 'min\_samples\_split': [2, 3], # The minimum number of samples required to split an internal node

# 'min\_samples\_leaf': [1, 2] # The minimum number of samples required to be at a leaf node

#}

# Set up the refined parameter grid based on initial findings (Hyperparameter Tuning 2)

# Wider range

# Include more options

# Wider range and None for unlimited depth

# Additional values

# Additional values

# Initialize a GridSearchCV object

# Fit the grid search to the data

# Print the best parameters and best score

# Random Forest Model with the best hyperparameters from GridSearchCV

# Call the function to plot Actual v Predicted

# Call the function to plot the residuals between Actual v Predicted