*University of Tehran*

**Professional and Specialized Training Office**

# Predicting Student Grades

**Final Project for Data Science in Python Course**

**Student Name: Shayan Ganji**

**Instructor: Dr. Mohsen Yazdinejad**

# Table of Contents

## Project Description

In machine learning, working with datasets that have numerous features but limited data points presents significant challenges. This project focuses on predicting student performance in a mathematics course using a dataset with diverse features that describe student demographics, family background, and academic history. Given the limited size of the dataset, our objective is to explore and compare different machine learning approaches to determine which techniques are most effective for this specific data setup.

Handling datasets with high-dimensional features but few observations can lead to issues such as overfitting and poor model generalization. Therefore, we aim to experiment with various preprocessing techniques, modeling strategies, and hyperparameter tuning methods to optimize performance on such datasets. This report will focus on analyzing the performance of different approaches and provide a brief explanation of the corresponding code implementations.

## Dataset Description

The dataset we are working with originates from a study that seeks to predict the final academic performance of secondary school students. It contains a variety of features, both numerical and categorical, that provide insight into each student's background and academic behavior. The dataset includes:

- **Numerical Features**: Attributes like student age, absences, and combined parental education levels, which provide continuous or count-based information.
- **Categorical Features**: Attributes like school type, gender, address type (urban or rural), and family support, which represent specific groups or categories.
- **Ordinal Features**: Some features, such as **travel time** to school and **study time**, are ordinal, meaning they represent ordered categories. While these are often treated as numerical in machine learning models, they do not carry a true numerical difference between the values.

The **target variable** in the training dataset is **G3**, which represents the final grade of students in mathematics. However, the test dataset lacks this target column, meaning we need to build models that accurately predict **G3** for the test set based on the available features.

# *Initial Data Overview*

Upon examining the dataset, several insights and patterns emerged that highlight potential challenges and areas of focus in our analysis. This section will explore key differences between the training and test datasets, as well as imbalances and skewed distributions within certain features.

## 1. Training vs. Test Data Differences

One of the primary observations is that the test dataset is significantly smaller than the training dataset. While the training dataset contains 359 records, the test dataset includes only 36 records. This discrepancy in size is a notable challenge for model evaluation, as a smaller test set increases the risk of bias in assessing model performance.

Additionally, the test and training datasets show similarities in terms of statistical properties such as **mean**, **standard deviation**, and **min/max values** for most numerical features. The similar distributions in both datasets indicate that the test set represents a consistent subset of the training data. For instance, the **mean age** of students in both datasets is close, and the distributions of **absences** and **health** are relatively aligned, though the test set exhibits slightly lower variance in certain features like absences.

## 2. Imbalanced Features

Several features in the dataset exhibit a high degree of imbalance, which could impact the model's ability to learn effectively:

- **School Type**: In both the training and test sets, a large majority of students belong to the 'GP' school (approximately 88% in training and 91.6% in the test set). This imbalance means the model will have limited information on students from the 'MS' school, potentially leading to biased predictions for this subgroup.
- **Parent Status (Pstatus)**: Nearly 90% of students in both the training and test datasets live with both parents, with only about 10% living apart. This could make it difficult for the model to learn differences in outcomes between students based on their parental cohabitation status.
- **Study Time**: A majority of students report a moderate amount of study time (between 2 to 5 hours per week), with over 50% in both datasets falling into this category. However, very few students study more than 10 hours per week, potentially limiting the model's understanding of how extensive study time affects final grades.

## 3. Ordinal Features Treated as Numerical

While features like **travel time** and **study time** are ordinal in nature (i.e., the values represent ordered categories), they are often treated as numerical data for the purposes of machine learning models. This simplification is common but can lead to inaccuracies if the model assumes that differences between categories are linear. For example, the difference between "1: less than 15 minutes" and "2: 15 to 30 minutes" for travel time may not be equivalent to the difference between "3: 30 minutes to 1 hour" and "4: more than 1 hour."

## 4. Distributions and Skewed Data

Several features show highly skewed distributions:

- **Failures**: The vast majority of students (over 78% in the training set) have not failed any classes, while very few students report multiple failures. This imbalance may result in a model that struggles to predict outcomes for students with high failure rates, as the data is heavily weighted toward students with no prior failures.
- **Alcohol Consumption**: Both **workday alcohol consumption (Dalc)** and **weekend alcohol consumption (Walc)** exhibit skewed distributions, with most students reporting very low consumption. Nearly 70% of students in the training set report the lowest level of alcohol consumption during workdays, while a slightly larger spread exists for weekend consumption.

## 5. Target Variable Distribution

The distribution of the target variable, **G3**, shows that a large proportion of students achieve grades between 8 and 13, with the mean around 10.34. However, a significant number of students score below 5, including some with a grade of 0. This creates a somewhat bimodal distribution, which could complicate the prediction task, as the model will need to account for both high and low outliers.

## 6. Lack of Severe Imbalance

Despite some noticeable imbalances, especially in **school type** and **parental status**, no extreme imbalances exist that would drastically skew model learning. Features such as **gender** and **famsize** are well-represented, ensuring that the model has a diverse range of data to learn from.

However, notable imbalances and skewness within certain features must be accounted for in preprocessing to ensure the model can generalize well to both high-performing and underperforming students

# Data Preprocessing

In this project, we began by selecting a baseline model and systematically explored various preprocessing techniques to optimize its performance. The **Linear Regression** model was chosen as the simplest baseline to start with. However, before applying such a model, even for a basic scenario like this, preprocessing is necessary.

## Encoding

A key challenge when working with categorical features is the need to transform them into numerical values that can be processed by machine learning algorithms like **Linear Regression**. For this, we used **Label Encoding**, one of the simplest and most commonly used encoding methods. This transforms categorical values into numerical representations without introducing additional complexity.

## Categorical Data Processing

Categorical data presents challenges since models like **Linear Regression** cannot handle non-numerical inputs. While more complex models like **CatBoost** internally handle categorical data, their use as a baseline is less appropriate due to their complexity. In contrast, simple linear models need explicit encoding before being trained on categorical features.

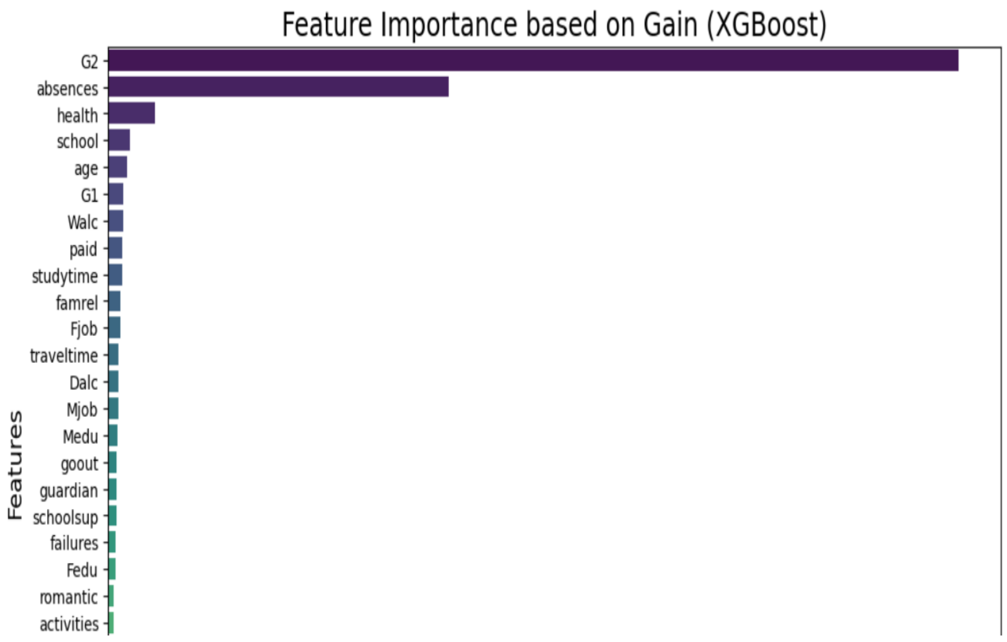
While **Label Encoding** was sufficient for the initial implementation, a more advanced approach such as **one-hot encoding** or **target encoding** could further improve model performance for categorical data. The next step was experimenting with a baseline linear model using label-encoded data, after which we evaluated different feature importance techniques to assess the contributions of different features.

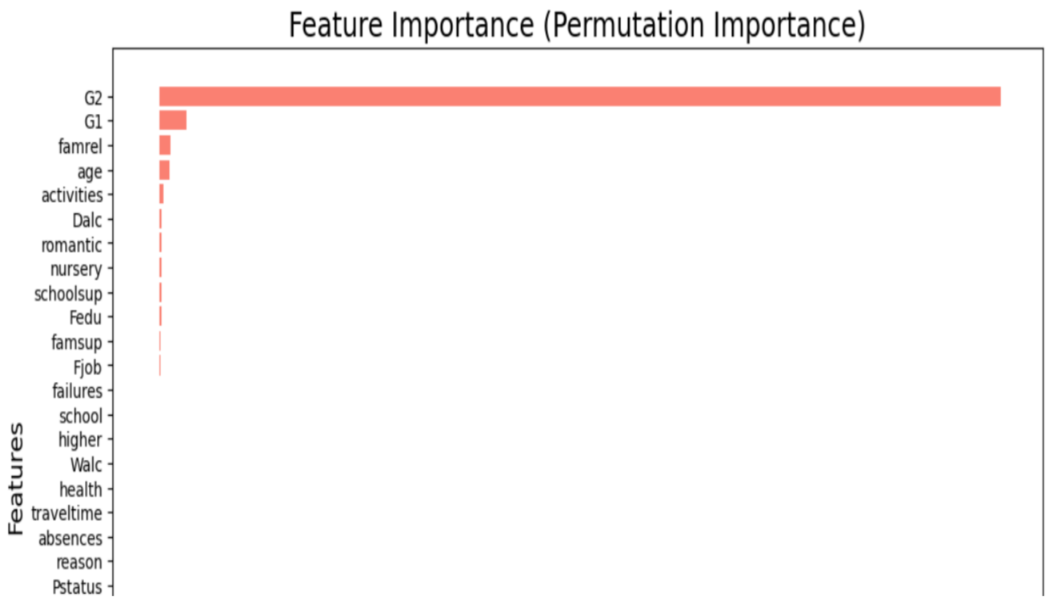
## Feature Importance Techniques

We used two different techniques to assess feature importance:

1. **Gain-based importance using XGBoost**: Gain represents the improvement in accuracy brought by a feature to the branches it is used in within the decision trees. Features with high gain contribute significantly to making accurate predictions.
2. **Permutation Importance using Scikit-learn**: Permutation importance measures the effect of randomly shuffling a feature's values on the model's performance. Features

## Analysis of Results





- Baseline RMSE: The initial RMSE of the baseline **Linear Regression** model was **1.969**, indicating room for improvement.

Feature Importance:

Incorporating both **gain-based importance** and **permutation importance** techniques helped us uncover which features contribute the most to model performance. Based on the findings:

- **G2** consistently emerged as the most significant feature, both in the XGBoost gain-based importance and permutation importance. This shows that **numerical features**—particularly those related to past academic performance—are critical for predicting future outcomes, which is our target variable **G3**.
- Additionally, features such as **G1** and **absences** also demonstrated moderate importance. These findings suggest that while categorical features are prevalent in our dataset, their overall contribution is relatively low compared to numerical ones.

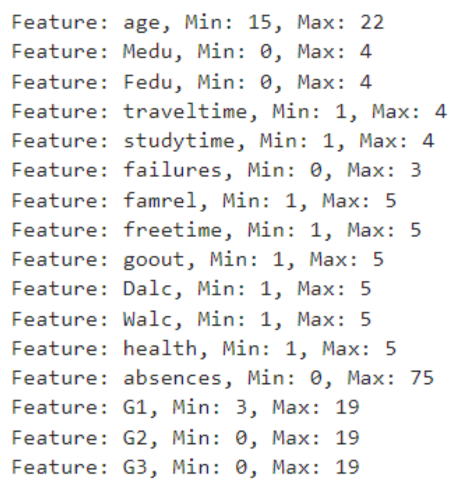
### Conclusion and Insights

Given that the most important features are overwhelmingly numerical, we can infer that while advanced models like **CatBoost** are specifically designed to handle categorical data, they may not necessarily be the optimal choice for this particular dataset. Although **CatBoost** excels at managing categorical features without explicit preprocessing, its benefits may be diminished when the dominant predictive features are numerical.

Moreover, the insights gained from the feature importance analysis suggest the need for thorough **feature engineering**. Many of the categorical features, especially those with minimal or negative contributions, could be refined, reduced, or engineered to enhance their influence in the model. This step will be crucial for improving the overall performance of the model and reducing noise introduced by less impactful features.

## Step 1: Outlier Management

Before diving into outlier management, we first examined the minimum and maximum values for all numerical features in the dataset (including ordinal features). The goal was to understand the natural range of values for each feature, as shown in the table.

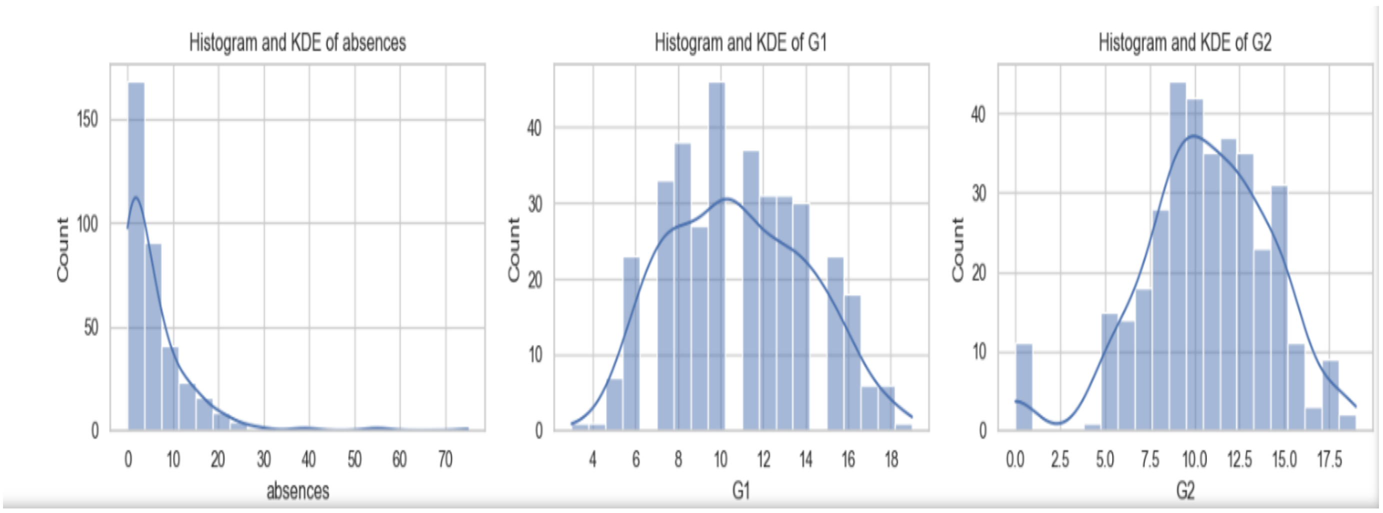


```
Feature: age, Min: 15, Max: 22
Feature: Medu, Min: 0, Max: 4
Feature: Fedu, Min: 0, Max: 4
Feature: traveltime, Min: 1, Max: 4
Feature: studytime, Min: 1, Max: 4
Feature: failures, Min: 0, Max: 3
Feature: famrel, Min: 1, Max: 5
Feature: freetime, Min: 1, Max: 5
Feature: goout, Min: 1, Max: 5
Feature: Dalc, Min: 1, Max: 5
Feature: Walc, Min: 1, Max: 5
Feature: health, Min: 1, Max: 5
Feature: absences, Min: 0, Max: 75
Feature: G1, Min: 3, Max: 19
Feature: G2, Min: 0, Max: 19
Feature: G3, Min: 0, Max: 19
```

Given that the data contains reasonable upper and lower bounds, careful consideration was necessary when managing outliers. For example, if the model were to identify the highest grades in the **G2** column as outliers, it could reduce the model's ability to generalize to students with high academic performance. This could lead to poor prediction performance for the test set, especially since the mean values tend to skew lower, making the preservation of high grades critical. Retaining such data can sometimes improve model generalization, as is the case in this project.

In our analysis, multiple techniques for managing outliers were explored. However, it's important to note that managing outliers doesn't always mean removing them entirely. Techniques like **capping** or **coercing** can be applied, where outliers are adjusted to the nearest acceptable value, rather than discarded.

Before proceeding with outlier detection, understanding the **distribution** of the data is crucial. Different methods of outlier detection are sensitive to distribution characteristics. For example, **Z-score** methods work best with normally distributed data. Below, we display the distributions of three key numerical features. Further distributions of other features can be seen in the provided Jupyter notebook.



## Outlier Detection Methods Tested

Several approaches were used to detect and manage outliers, and the following results were obtained for the training data:

```
RMSE after IQR Outlier Management: 1.5349581129139547
RMSE after Z-Score + IQR Outlier Management: 1.904787134271719
RMSE after Isolation Forest Outlier Management: 1.8887450276846134
RMSE after LOF Outlier Management: 1.926969058542631
RMSE after Elliptic Envelope Outlier Management: 1.270010217030492
```

The **Elliptic Envelope** method provided the best performance in terms of RMSE, achieving 1.2700, significantly outperforming other methods. The reason behind this can be attributed to the fact that Elliptic Envelope identified only a small number of outliers (just 3), preserving most of the data. This reinforces the initial assumption that our dataset contains very few true outliers, and retaining them may be beneficial for model accuracy.

The **Elliptic Envelope** method works by fitting an elliptical boundary around the data and identifying points that fall outside of this ellipse as outliers. Unlike more aggressive techniques like **Z-score** or **IQR** (which assume a normal distribution or simple range limits), Elliptic Envelope adapts better to the overall structure of the dataset, making it a more effective solution in this case.

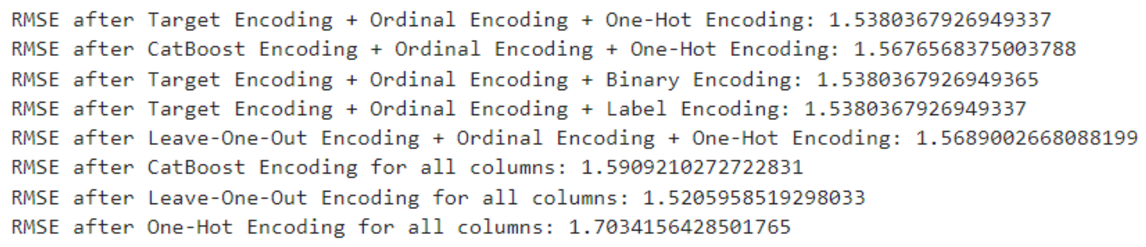
In conclusion, the careful preservation of data during outlier management proved valuable, and this insight guided the decision-making process in subsequent modeling steps. Though multiple techniques were tested, the subtle approach of Elliptic Envelope, which retains most of the data, is what ultimately yielded the best results. But; in the final model, we employed strategies to limit the impact of outliers rather than removing them entirely, ensuring the model's ability to generalize well to both high and low-performing students.

## Step 2: Encoding

Selecting the appropriate encoding method is influenced by several factors, most notably the model being used. In this project, we tested different encoding techniques using a simple linear regression model as a baseline. It's crucial to note that while the encoding methods were evaluated with a consistent model (linear regression), this does not imply that the best-performing encoding in this context would universally be the optimal choice across different models. The results, however, do provide insight into which encodings might be more suitable for linear models.

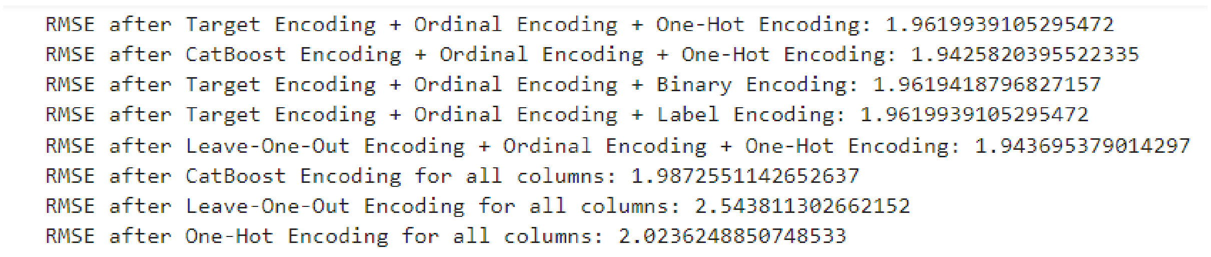
In one of our experiments, we deliberately altered the preprocessing order, performing encoding before outlier management, which is not a typical practice. The results from this reordered pipeline demonstrated a significant drop in model performance, emphasizing the importance of following a logical preprocessing sequence. Although the columns affected by outlier management and encoding were different, there was a clear interaction between these steps, reinforcing that encoding cannot be assessed independently from other data preprocessing tasks.

**Data Preprocessing in an Organized Manner:**



```
RMSE after Target Encoding + Ordinal Encoding + One-Hot Encoding: 1.5380367926949337
RMSE after CatBoost Encoding + Ordinal Encoding + One-Hot Encoding: 1.5676568375003788
RMSE after Target Encoding + Ordinal Encoding + Binary Encoding: 1.5380367926949365
RMSE after Target Encoding + Ordinal Encoding + Label Encoding: 1.5380367926949337
RMSE after Leave-One-Out Encoding + Ordinal Encoding + One-Hot Encoding: 1.5689002668088199
RMSE after CatBoost Encoding for all columns: 1.5909210272722831
RMSE after Leave-One-Out Encoding for all columns: 1.5205958519298033
RMSE after One-Hot Encoding for all columns: 1.7034156428501765
```

**Data Preprocessing in a Disorganized Manner:**



```
RMSE after Target Encoding + Ordinal Encoding + One-Hot Encoding: 1.9619939105295472
RMSE after CatBoost Encoding + Ordinal Encoding + One-Hot Encoding: 1.9425820395522335
RMSE after Target Encoding + Ordinal Encoding + Binary Encoding: 1.9619418796827157
RMSE after Target Encoding + Ordinal Encoding + Label Encoding: 1.9619939105295472
RMSE after Leave-One-Out Encoding + Ordinal Encoding + One-Hot Encoding: 1.943695379014297
RMSE after CatBoost Encoding for all columns: 1.9872551142652637
RMSE after Leave-One-Out Encoding for all columns: 2.543811302662152
RMSE after One-Hot Encoding for all columns: 2.0236248850748533
```

## Why Does Changing the Order Impact Results?

The notable impact of changing the preprocessing order stems from how outliers can influence encoded features. For example, while categorical columns are handled separately during encoding, outliers in numerical columns might skew relationships across the dataset. When outlier management is delayed, it can affect the encoded values, especially in methods like target encoding, where categorical features are transformed based on the target variable. This highlights the importance of conducting encoding after outlier management to ensure that both numerical and categorical data are accurately represented.

## Comparison of Encoding Methods

When dealing with categorical features, selecting the right encoding method is crucial, as it can significantly affect model performance. Here, we compare several commonly used encoding techniques and their implications:

1. **Target Encoding**: Target encoding replaces categories with the mean target value for each category. This method can be highly effective, especially for high-cardinality features, but it carries the risk of **overfitting**, particularly when certain categories have very few samples. In such cases, the encoded value can be overly influenced by outliers or noise in the data, leading to poor generalization on the test set. For instance, features like **school** or **address**, with only two categories, are prone to overfitting since one dominant category may skew the target mean.

2. **Leave-One-Out Encoding**: This method addresses the overfitting issue by calculating the mean target for each category, excluding the current sample. By removing the sample

from its own category calculation, the risk of overfitting is reduced. This makes Leave-One-Out encoding more robust for small datasets or categories with limited samples, as it prevents the model from overly memorizing the target variable for specific instances.

3. **One-Hot Encoding**: One-hot encoding is a straightforward approach where each category is represented by a separate binary column. While it prevents overfitting by avoiding the use of target information, it can dramatically increase the dimensionality of the dataset, particularly for features with many categories. This increase in dimensionality can lead to inefficiencies, especially for models sensitive to feature size, such as linear models.
4. **CatBoost Encoding**: This encoding method, used in **CatBoost** models, automatically handles categorical data by calculating aggregated statistics on-the-fly. While this method simplifies the handling of categorical data, it's more suited to the **CatBoost** model itself, which can internally manage categorical features. Using this encoding with other models may not yield the same performance benefits and could introduce additional complexity without significant improvement.

**Key Challenges in Encoding**

- **Overfitting with Target-Based Encodings**: Both **Target Encoding** and **Leave-One-Out Encoding** are vulnerable to overfitting, particularly for features with a small number of categories or imbalanced distributions. For example, in our dataset, the features **school** and **address** each have two categories, with one dominant category. The risk here is that target-based encodings can heavily rely on the training data, reducing model generalizability.
- **Dimensionality with One-Hot Encoding**: While **One-Hot Encoding** avoids overfitting, it can dramatically increase the feature space, especially when applied to high-cardinality features. This can be problematic for algorithms that struggle with high-dimensional data, making it a less optimal choice for models like linear regression.

**Combining Encoding Methods**

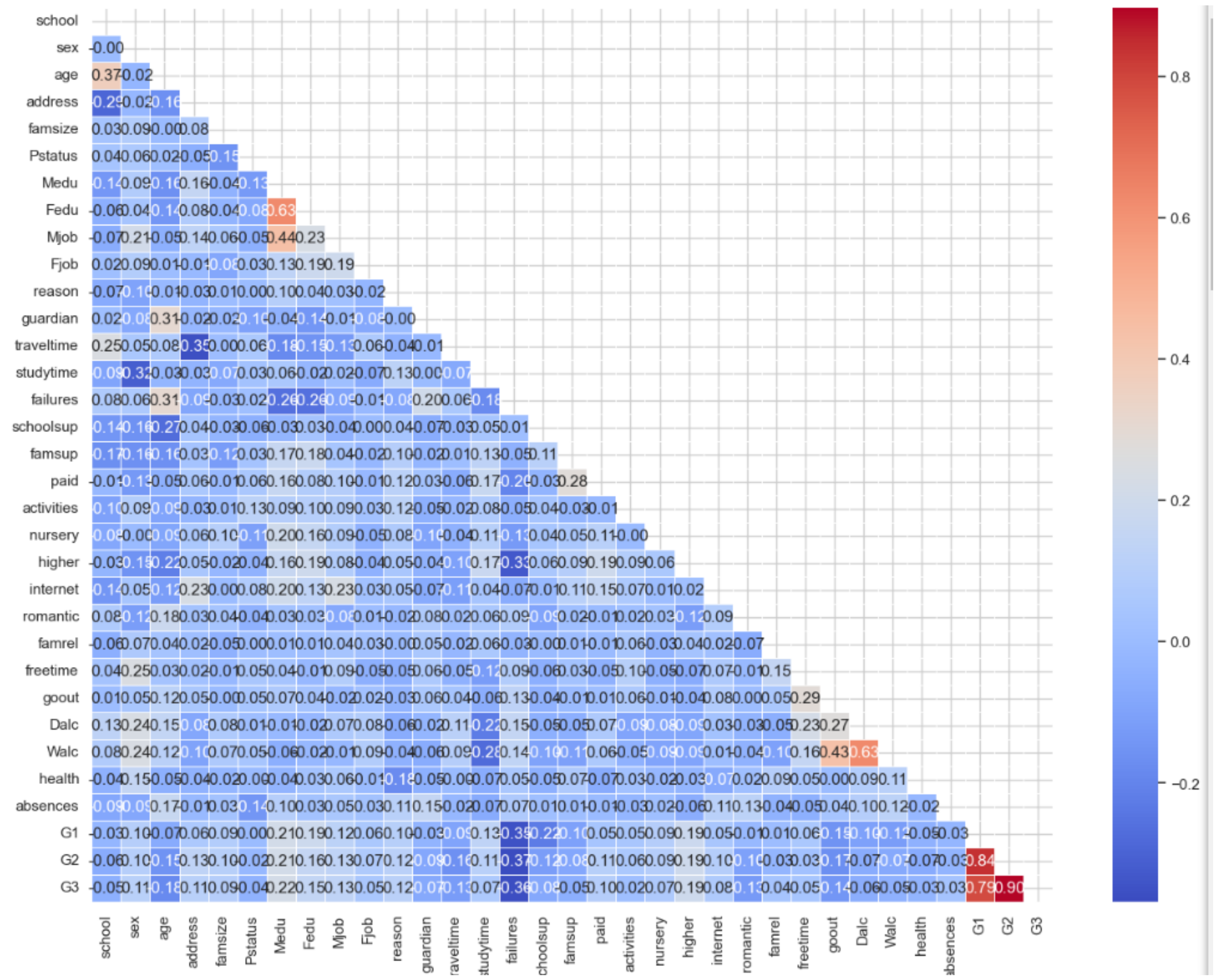
Using a combination of encoding techniques can be advantageous but comes with its own set of challenges. On one hand, hybrid encoding methods allow us to leverage the strengths of multiple techniques, for instance, using **Target Encoding** for high-cardinality features while applying **One-Hot Encoding** to low-cardinality features. This approach balances the need for model flexibility with the prevention of overfitting.

On the other hand, combining encoding methods increases the complexity of the preprocessing pipeline and the dimensionality of the data, which can adversely affect certain models. Linear models, for example, may struggle with the increased number of features resulting from **One-Hot Encoding** when used alongside other encodings. Additionally, the interaction between encoded features and other preprocessing steps (like outlier management) must be carefully managed to ensure consistent results.

## Visualization and Feature Engineering

### Correlation Analysis Overview

In this section, we conducted a correlation analysis to identify the relationships between different features in our dataset. A correlation matrix was plotted to visualize the strength and direction of linear relationships between pairs of features. This analysis helps in understanding which features are strongly associated, both positively and negatively, and can guide feature selection or engineering decisions.



The heatmap presented above provides a visual representation of these correlations, with darker shades indicating stronger correlations, either positive or negative. The sns.heatmap function was used, with settings that enhanced the clarity of the plot, particularly by masking the upper triangle to reduce redundancy. This allows us to focus on key relationships without distractions.

**Key Findings from Correlation Analysis**

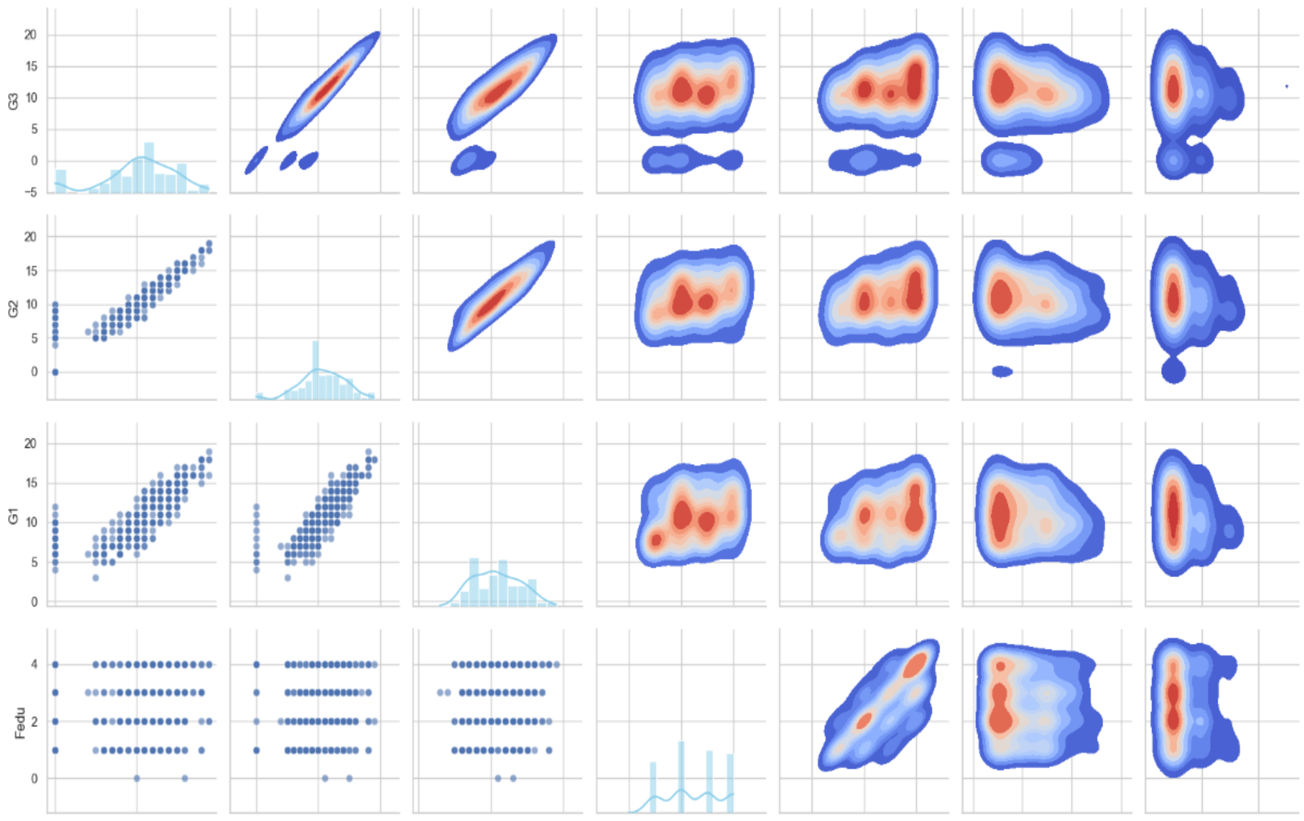
- **Top Positive Correlations**:
    - **G3 and G2 (0.897)**: This very strong positive correlation between the final grade (**G3**) and the second-period grade (**G2**) is expected, as the grades are highly sequential and likely influence each other significantly. This indicates that students' performance in earlier periods is a strong predictor of their final grade.
    - **G2 and G1 (0.843)**: Similarly, the first-period grade (**G1**) is strongly correlated with the second-period grade (**G2**). The performance across the three grading periods is thus highly interlinked, reflecting the consistency in student performance throughout the academic year.
    - **G1 and G3 (0.786)**: There is also a strong correlation between the first and final grades, reinforcing the idea that earlier academic performance is a key indicator of final outcomes.
    - **Fedu and Medu (0.627)**: The educational background of both parents (**Fedu** and **Medu**) also exhibits a strong positive correlation, which may reflect socioeconomic factors where higher education levels of one parent are likely associated with the education level of the other.
    - **Walc and Dalc (0.627)**: The strong positive correlation between **weekend alcohol consumption (Walc)** and **weekday alcohol consumption (Dalc)** suggests that students who consume alcohol during the weekdays are also likely to drink on weekends.
- **Top Negative Correlations**:
    - Interestingly, no strong negative correlations (below -0.5) were found in this dataset, indicating that there are no features that exhibit a significant inverse relationship with others. This suggests that the features included in this dataset are more complementary or neutral in their relationships.

**Insights for Feature Engineering**

The high correlations between the three grades (**G1**, **G2**, and **G3**) confirm that these variables hold significant predictive power for student performance, making them crucial features in any predictive model. However, their high inter-correlation might also lead to redundancy, indicating the potential for feature engineering approaches like creating composite variables (e.g., **G1_G2**) to reduce multicollinearity without losing predictive power.

Additionally, the strong correlation between parental education levels (**Fedu** and **Medu**) suggests that including only one of these features or combining them into a single metric could simplify the model without sacrificing much information. Similarly, the strong association between **Walc** and **Dalc** hints that combining these two into a single measure of alcohol consumption could enhance model efficiency.

## Pair Grid Analysis Overview



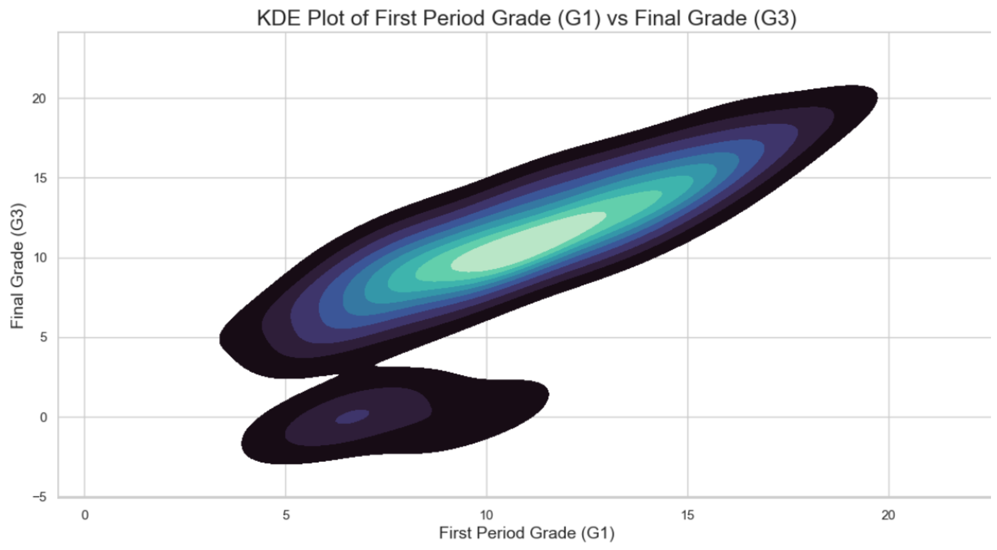
The Pair Grid visualization offers a multi-faceted view of the relationships between the selected features, including Top Positive Correlation **G1**, **G2**, **G3**, and **Fedu** (and more features outside the displayed portion).

The plots are divided into three parts:

1. **Diagonal Plots (Distributions)**:
    - The diagonal contains histograms and KDE plots, displaying the distribution of each feature. For example, grades (G1, G2, G3) show a roughly normal distribution with slight skewness, reflecting the overall spread of student performance. Parental education levels (Fedu) exhibit a more discrete distribution, as expected from categorical-like data.

2. **Upper Triangle (Density Estimates)**:
   - The upper triangle shows KDE (Kernel Density Estimation) plots, which help visualize the joint distribution between feature pairs. The strong density overlaps between **G1**, **G2**, and **G3** confirm their high positive correlations. The tighter and more concentrated ellipses in these plots signal strong relationships.
3. **Lower Triangle (Scatter Plots)**:
   - Scatter plots in the lower triangle give a raw visualization of the correlation between features. The linear patterns between the grades further emphasize their interdependence. **Fedu** (parental education) has a more dispersed relationship with grades, indicating weaker correlations with student performance.

## Analysis of One of the Upper Triangle Plots



The Kernel Density Estimate (KDE) plot effectively visualizes the joint distribution of the grades G1 (First Period) and G3 (Final Grades). The key insight is the existence of two distinct regions of density, which suggests some form of differentiation in the data.

**Two Clusters of Data - Explanation Refinement:**

- **Cluster 1 (Higher Grades)**: This cluster represents students who have scored consistently higher in both **G1** and **G3** (Grades between 10 to 20). The smooth and elongated oval shape indicates a strong positive correlation between these two grades.

    - **Interpretation**: These students likely maintained a consistent level of performance throughout the academic year.

- **Cluster 2 (Lower Grades)**: The second, smaller density region represents students who performed poorly, with grades in the range of **0 to 10** for **G1** and **G3**. The elliptical spread is more stretched, indicating a more variable performance pattern in this group.
    - **Interpretation**: This cluster represents students who either started weak and continued to perform poorly or students who might have struggled in both assessments.

**Key Insights:**

- **Bimodal Distribution**: The presence of two separate clusters suggests that the student population is divided into two distinct performance categories, with minimal overlap.
- **Non-linear Correlation**: The KDE plot shows an **elliptical shape**, which indicates a non-linear relationship between G1 and G3. This suggests that improvements in G1 do not always result in a proportional improvement in G3, and vice versa.
- **External Factors**: The separation into two groups could be influenced by various external factors such as **family support, socioeconomic status**, or **extra-curricular engagement**, which are likely contributing to the divergence in student performance.

**Solutions:**

To address the challenge of distinguishing between these two distinct student groups and predicting future performance, two potential approaches are considered: using **non-linear models** or applying **clustering techniques**.

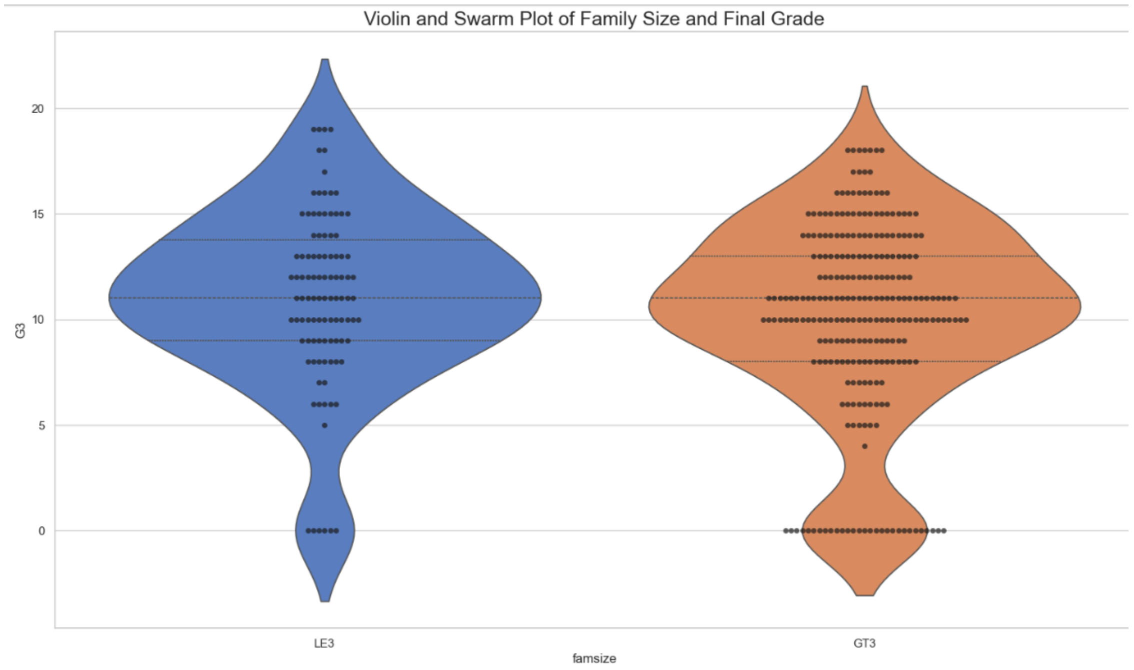
1- **Non-linear Models**: While non-linear models can capture complex relationships between variables like G1 and G3, they often pose a high risk of **overfitting** when applied to datasets with limited size. In such cases, the model may fail to generalize well to new data, reducing its predictive power.

2- **Clustering**: Given the bimodal nature of the data, clustering proves to be a more suitable method for this dataset. It allows for grouping students based on similar performance patterns without the risk of overfitting inherent in non-linear models. As a result, clustering can effectively identify distinct performance groups and help in designing targeted educational interventions.

In this project, the final decision was made to implement clustering techniques due to their robustness in handling such challenges, avoiding the pitfalls of overfitting that might arise with non-linear models.

## External Factors

### 1- Family Size and Final Grade



As mentioned earlier in the Key Insights section, additional factors could contribute to the clustering of student grades. To explore this further, we examined the influence of family size (famsize) on students' final grades (G3). The Violin and Swarm Plot of Family Size and Final Grade shown above visualizes the distribution of grades based on family size, split into two categories: LE3 (families with 3 or fewer members) and GT3 (families with more than 3 members).

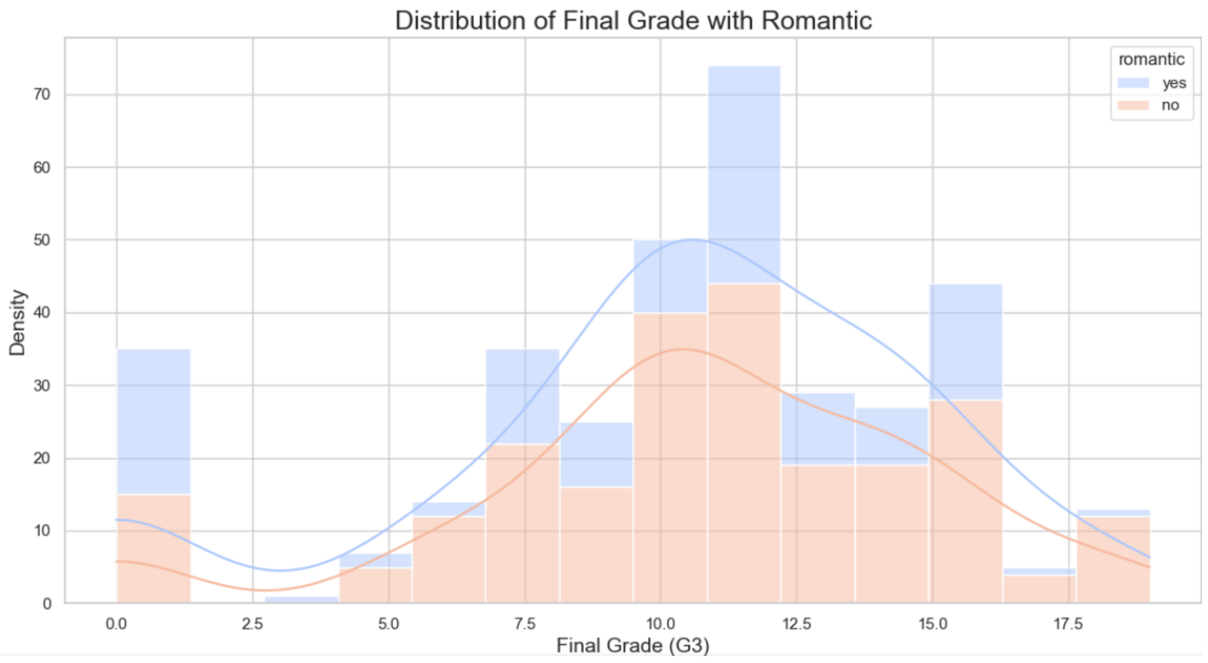
This visualization reinforces the hypothesis regarding the presence of clusters in student performance. The violin plot helps us observe the density and distribution of grades, while the swarm plot highlights individual data points, providing clarity on how family size might influence academic performance.

**Observations from the Plot:**

1. **LE3 (Families with 3 or fewer members)**:
   - This group shows a more spread-out distribution of grades, with students achieving both **high and low** scores. The density is more evenly distributed across different grade ranges.
   - There is a **moderate concentration** of students scoring between **10 to 15**.
2. **GT3 (Families with more than 3 members)**:
   - Students from larger families exhibit a more **compact distribution** of grades, with the majority scoring in the **10 to 15** range.
   - Interestingly, the lower tail of the distribution extends towards **0**, indicating that students from larger families may be more likely to score lower grades.

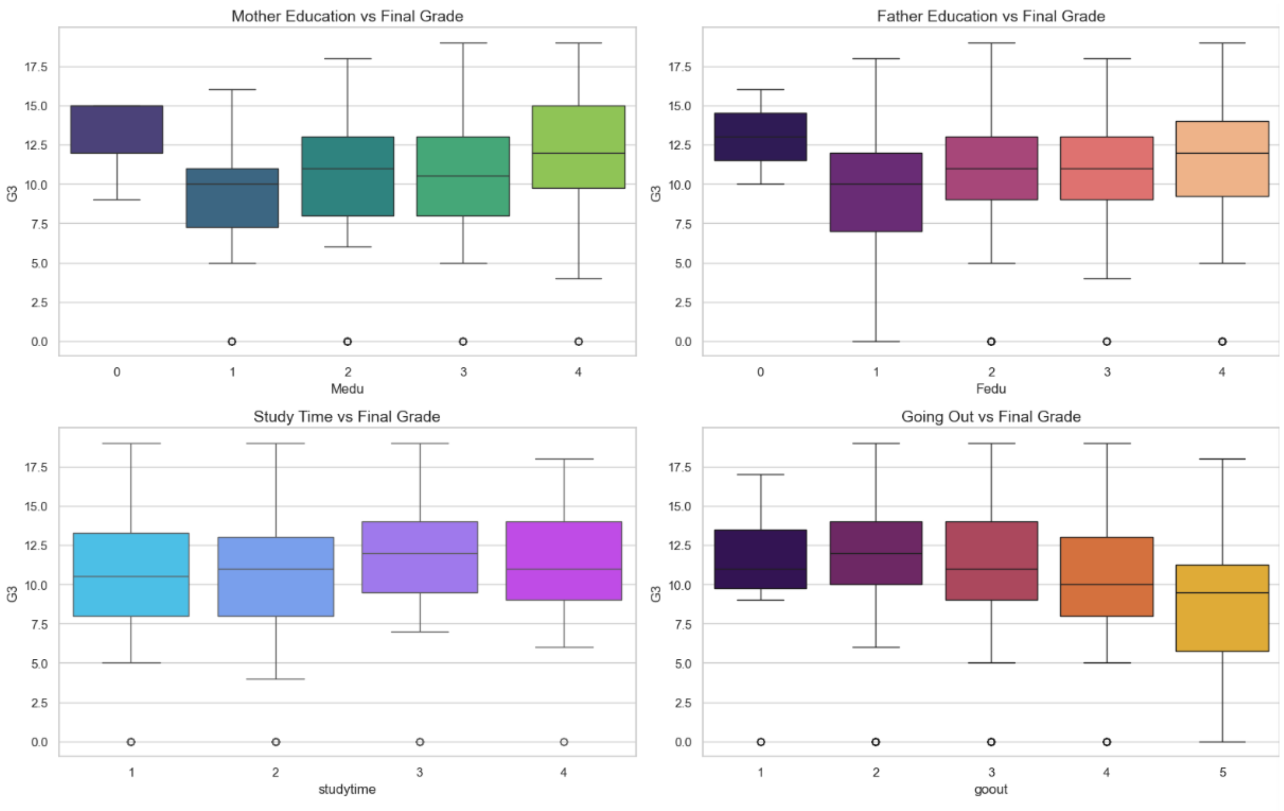
This violin and swarm plot aligns with our earlier insights about the clustering of student grades. It indicates that family size is a potential factor influencing the observed clusters, reinforcing the need to consider external variables when analyzing academic performance patterns.

## 2- Distribution Analysis (Romantic and G3)



This plot suggests that students in romantic relationships tend to cluster around average grades and may be less likely to achieve the highest final grades. Conversely, students without romantic relationships seem to have a slightly better representation in higher grade ranges. However, the difference is subtle and requires further investigation into other possible contributing factors to performance. Both groups, romantic and non-romantic, have students scoring below 5, with a slightly higher proportion among students in relationships, though this is not overwhelmingly different.

## 3- Analysis of Boxplots: Relationship Between Features and G3

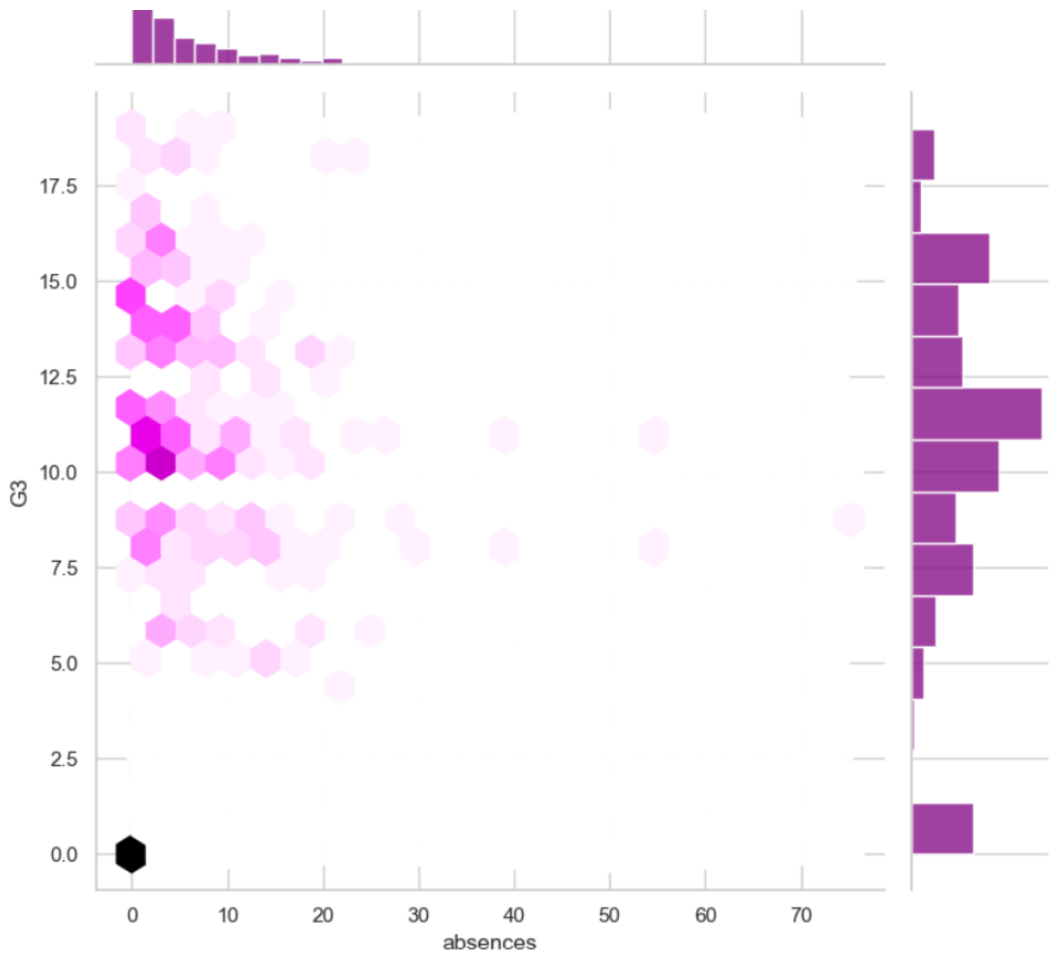


**Parental Education:** Both mother’s and father’s education levels positively correlate with final grades, with mother’s education showing a slightly stronger influence. Interestingly, students whose parents have the lowest education level (Medu/Fedu = 0) tend to perform better than those whose parents are at the first level of education (Medu/Fedu = 1). This suggests that the relationship between parental education and student performance is not strictly linear, though the trend becomes more linear from level 2 onwards.

**Study Time:** Increased study time generally leads to higher grades, although the spread of grades indicates that study time alone is not the sole determinant of academic success.

**Socializing(goout):** Excessive socializing appears to have a negative impact on academic performance, with students who go out more frequently scoring lower on average.

## 4- Hexbin Joint Plot Absences vs Final Grade (G3)



A joint plot combines two types of visualizations: one that displays the relationship between two variables (in this case, absences and final grade), and histograms of each variable along the x and y axes to show the distributions individually. In this plot, a hexbin plot (hexagonal binning plot) is used to visualize the density of data points, which is especially useful when dealing with a large number of data points or overlapping values. The density is indicated by the color intensity, with darker areas representing a higher concentration of data points.

There is a noticeable cluster of students with lower absences (fewer than 10) and a wide range of final grades, from very low (0) to high (17.5). However, as the number of absences increases beyond 10, final grades tend to cluster more in the lower range, with very few students achieving higher grades (above 12.5).

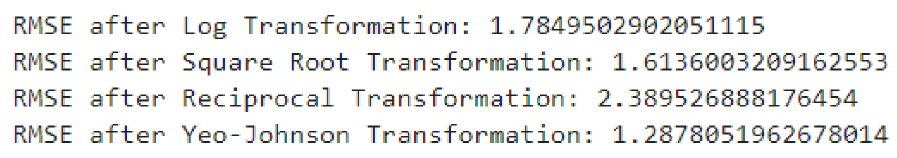
**Black Hexagonal Region** (Zero Absences, Zero Final Grade):

- The **black hexagon** on the bottom-left corner represents a dense concentration of students who have both **0 absences** and a final grade of **0**.
- After further analysis, it was discovered that **35 students** fall into this category, which is unusual. These students attended all classes but still received the lowest possible grade.

In the future analyzes and in the final modeling, this black region should be considered

## Step 3: Transformation

Several transformation methods were tested, and their performance was measured using the **Root Mean Squared Error (RMSE)**.



```
RMSE after Log Transformation: 1.7849502902051115
RMSE after Square Root Transformation: 1.6136003209162553
RMSE after Reciprocal Transformation: 2.389526888176454
RMSE after Yeo-Johnson Transformation: 1.2878051962678014
```

Below is an interpretation of why certain transformations led to better results and why others performed poorly, along with an additional focus on the specific characteristics of the dataset.

- **Log Transformation**
  The log transformation works well when the data has skewness, particularly when large values dominate. It compresses the range of the data, reducing the effect of extreme values. In this dataset, however, the log transformation did not significantly improve performance because most numeric features did not exhibit extreme skewness. While it helped slightly with normalization, it did not fully address the underlying variance, which limited its effectiveness. Furthermore, since log transformations tend to struggle with zero or negative values (even after adjusting by adding a constant), the error remained relatively high. Given that 35 students in the dataset simultaneously received a grade of zero and had zero absences, applying a log transformation without an adequate constant would introduce many negative or undefined values, which further reduced the efficacy of this method.
- **Square Root Transformation**
  The square root transformation often works well for moderately skewed data by

reducing the variance more effectively than log transformations. In this case, the square root method provided better results than the log transformation because it could handle the smaller variations in the data distribution more effectively. However, it was not ideal for handling negative or very small values, which is why its performance, while improved, still left some residual error. The fact that there are numerous entries close to zero in this dataset (due to students with both zero grades and zero absences) means that the square root transformation struggled to effectively differentiate between these values, resulting in suboptimal performance.

- **Reciprocal Transformation**
  The reciprocal transformation tends to amplify the effect of small values while compressing large ones. However, in this scenario, this approach resulted in a poor performance because it disproportionately shrank larger values, distorting the relationships within the data. The reciprocal transformation is typically suited to data where there is a need to heavily downscale high values and where small values play a crucial role. This was not the case in our dataset, as the grades and other numeric features were relatively balanced without extreme disparities. Additionally, the abundance of data points close to zero in the dataset would make the reciprocal transformation unsuitable because it would drastically amplify these values, skewing the model's understanding of the data structure.

- **Yeo-Johnson Transformation**
  The **Yeo-Johnson transformation** proved to be the best-performing method in this context, particularly because of its ability to handle both positive and negative values without requiring constant adjustments. The dataset presents a unique challenge: 35 students have both a grade of zero and zero absences, creating a situation where many data points are very close to zero. If a constant is not added in other transformation methods, these values could turn negative or cause distortions in the analysis.

Yeo-Johnson's capability to adapt without requiring arbitrary constants, made it a superior choice in this situation.

## Step 4: Scaling

For the final scaling decision, we chose **Robust Scaling** over other scaling methods such as standard or min-max scaling. The rationale for this decision is grounded in the nature of the data and the results observed during the transformation phase:

- During our analysis, we identified unusual patterns in the data, such as students attending all classes but receiving a zero grade. These patterns were not traditional outliers when viewed on a per-column basis (like min or max values), but they represented unusual relationships between features.

- While **Elliptic Envelope** and other outlier management methods improved the model’s performance on the training data by removing extreme values, they led to **overfitting** and consequently worsened test data performance. This suggested that handling outliers too aggressively made the model overly sensitive to the training data's structure, thus reducing its generalization ability.
- **Robust Scaling** was selected because it relies on the **median** and **interquartile range (IQR)** for scaling, making it inherently resistant to the influence of extreme values or anomalies. This approach allows us to model the data more realistically, ensuring that any unusual observations (like those mentioned earlier) do not disproportionately affect the scaling process. It strikes a balance between preserving data fidelity and mitigating the effect of outliers, which aligns with the goal of maintaining model accuracy without sacrificing generalization

# *Modeling*

## Step 1: Strategy Selection and Interpretation of Results

Given the nature of the dataset and its small dimensions, we refrained from using deep and complex models like neural networks, as these models generally require larger datasets for optimal performance. Instead, simpler and more interpretable models were chosen, which are more suitable for small datasets and help avoid overfitting.

**Feature Engineering Strategy**

Through extensive testing and experimentation, we observed that manual feature engineering—specifically manual feature aggregation—yields the best results for this dataset. Although we applied **Principal Component Analysis (PCA)** with various configurations, none of the PCA settings performed as well as manual adjustments.

**Key Benefits of Manual Feature Aggregation:**

1. **Preserving Specific Feature Information:** By manually aggregating features with high correlations (e.g., **Fedu** and **Medu**, **Walc** and **Dalc**, **G2** and **G1**), we ensure that critical information from each feature is retained. For example, merging **Fedu** (Father’s

education) and **Medu** (Mother's education) into a new feature like **parent_edu_level** provides a meaningful and interpretable feature that directly benefits prediction.

2. **Improved Model Interpretability:** Manually aggregated features contribute to better model interpretability. For instance, creating features like **parent_edu_level** or **alcohol_consumption** allows the model to work with more semantically understandable inputs, enhancing the transparency of predictions.
3. **Avoiding Complexity from Dimensionality Reduction:** While PCA reduces dimensionality, it can obscure the relationship between original features and new components, complicating model interpretation. In contrast, manual aggregation does not introduce this complexity and ensures that newly created features are directly linked to meaningful original features.
4. **Preventing Loss of Key Information:** Unlike PCA, which may discard information that isn't linearly significant, manual feature aggregation retains crucial data. By strategically merging features based on domain knowledge and correlation analysis, we can preserve essential information.

In conclusion, Using **PCA** in this context could reduce model accuracy because important information might be lost in the dimensionality reduction process. On the other hand, manual feature aggregation, based on correlation analysis and domain knowledge, maintains key information and improves overall model accuracy.

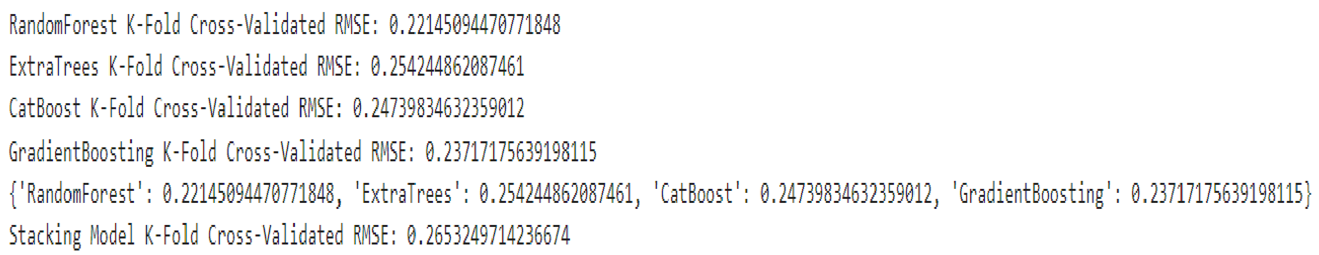
## Step 2: Model Evaluation and Discussion

In this phase, we are merely exploring some **alternative modeling approaches** that were tested throughout the project. These are not the final models used in our final solution, but they provide valuable insights into how different algorithms perform on this dataset. We will revisit and discuss what these insights mean in more detail in the final section under the **Final Model** heading.

**Data Leakage and Comparison of Two Approaches**

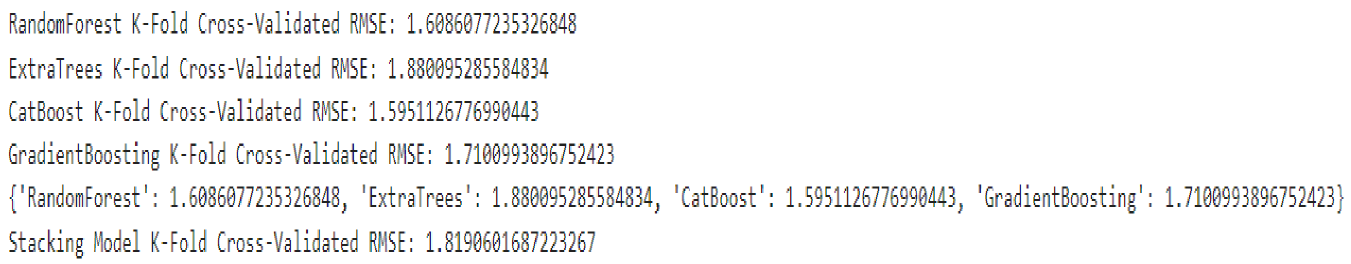
An important distinction between the two modeling approaches concerns **data leakage**. While true data leakage is not possible with the test data (as it lacks the target variable), leakage within the training-validation split can affect the validity of our results.

In **the first approach**, we introduced potential data leakage by preprocessing steps before separating the target variable. This could unintentionally allow the target variable to influence the process, which led to **artificially lower RMSE values**.



```
RandomForest K-Fold Cross-Validated RMSE: 0.22145094470771848
ExtraTrees K-Fold Cross-Validated RMSE: 0.254244862087461
CatBoost K-Fold Cross-Validated RMSE: 0.24739834632359012
GradientBoosting K-Fold Cross-Validated RMSE: 0.23717175639198115
{'RandomForest': 0.22145094470771848, 'ExtraTrees': 0.254244862087461, 'CatBoost': 0.24739834632359012, 'GradientBoosting': 0.23717175639198115}
Stacking Model K-Fold Cross-Validated RMSE: 0.2653249714236674
```

In contrast, **the second approach** avoided data leakage by separating the target variable from the beginning of the preprocessing steps. As a result, the performance metrics from this method provide a more **accurate reflection of real-world performance**. Although the RMSE scores were slightly higher, this approach more closely mirrors how the model would perform on unseen data, making it a more reliable approach.



```
RandomForest K-Fold Cross-Validated RMSE: 1.6086077235326848
ExtraTrees K-Fold Cross-Validated RMSE: 1.880095285584834
CatBoost K-Fold Cross-Validated RMSE: 1.5951126776990443
GradientBoosting K-Fold Cross-Validated RMSE: 1.7100993896752423
{'RandomForest': 1.6086077235326848, 'ExtraTrees': 1.880095285584834, 'CatBoost': 1.5951126776990443, 'GradientBoosting': 1.7100993896752423}
Stacking Model K-Fold Cross-Validated RMSE: 1.8190601687223267
```

**K-Fold Cross-Validation**

Given the small dataset, we applied **K-Fold Cross-Validation** to ensure robust evaluation. By splitting the data into multiple subsets and training on different portions, K-Fold helps mitigate overfitting and provides a more reliable estimate of the model's generalization ability. This method is particularly suited to small datasets, offering a better performance metric than a simple train-test split.

**Model Performance Comparison**

1. **Random Forest**, **Extra Trees**, and **Gradient Boosting** are all tree-based ensemble models, but each has distinct mechanisms:
    - **Random Forest** averages predictions from many decision trees, resulting in lower variance and improved generalization.
    - **Extra Trees** adds randomness to the split selection process, which sometimes increases error, as reflected in the higher RMSE.
    - **Gradient Boosting** builds trees sequentially, each correcting errors made by the previous tree, making it more suited for capturing complex relationships but potentially overfitting without proper tuning.

2. **CatBoost** performed consistently well, both in this phase and earlier in the project. One of its main advantages is its ability to handle categorical features natively without the need for preprocessing. Interestingly, the results show that **CatBoost performs best when no external preprocessing** (e.g., encoding, scaling) is applied, emphasizing its robustness for datasets with high proportions of categorical data. This insight was anticipated in our earlier discussions and is reaffirmed here.
3. **Stacking Regressor**, which combines multiple models, did not outperform the individual models. This may be attributed to the small size of the dataset and limited diversity in the base models' predictions, reducing the potential benefit of combining them.

In this section, we explored various models using **K-Fold Cross-Validation**, which is especially important in small datasets. We also highlighted the importance of correct data handling, where avoiding **data leakage** led to more realistic results. **CatBoost**, with its native handling of categorical features, was particularly effective when used without additional preprocessing, further confirming its suitability for datasets like ours. In the **Final Model** section, we will dive deeper into how these insights guide our final model selection and optimization.

## Step 3: Final Model

In this section, we will explore the final model developed for this project. Unlike the alternative models discussed earlier, this final model represents the optimal approach after extensive experimentation with feature engineering and model selection. This phase integrates insights gained from previous stages, and we will delve deeper into why certain decisions were made to achieve robust performance. Furthermore, we will examine how this final model effectively balances simplicity and accuracy using principles like **Occam's Razor**.

### Feature Engineering

The most critical aspect of this final model lies in the **feature engineering** techniques applied early in the process. The key difference here, compared to previous models, is the incorporation of **clustering** alongside feature merging.

1. **Merging Highly Correlated Features**: Columns with high correlation, such as Fedu and Medu (parental education), or Walc and Dalc (alcohol consumption), were merged to combat **multicollinearity**. Multicollinearity occurs when independent variables are highly correlated, leading to instability in regression models. Merging these variables simplifies the model and enhances interpretability without losing significant information.

2. **Clustering with K-Means**: Using the clustering technique in combination with **K-Means** added valuable dimensions to the dataset. We grouped students into clusters based on their first-term and second-term grades (G1, G2), as well as parental education and alcohol consumption. The rationale for this clustering approach comes from the 2D KDE plots we explored earlier, where we observed non-linear relationships between the grades and these features. This clustering helped segment the data and improved the model's ability to capture underlying patterns, which is especially crucial for a linear model like **Lasso Regression**. By introducing clusters, we enable the linear model to better handle non-linear relationships in the data.
3. **Weight Assignment**: Another significant addition to the final model is the adjustment of sample weights. As discussed earlier, students with **zero absences** and **zero final grades** were found to exhibit unusual behavior in the data. To prevent the model from being biased by these observations, we assigned them **half-weight**. This ensures that these data points still contribute to model training but do not disproportionately influence predictions.

**Encoding and Transformation**

The final model uses **Label Encoding** for categorical variables instead of more complex encoding methods like One-Hot or Target Encoding. The rationale here is to keep the model simple, as overly complex encoding methods could potentially lead to overfitting, particularly with such a small dataset. Label Encoding offers an efficient and straightforward way to handle categorical data while minimizing complexity.

The use of **Yeo-Johnson Transformation** was applied to numerical columns like Fedu_Medu, Walc_Dalc, and G1_G2, aiming to **normalize** the data. This transformation helps make the data more Gaussian-like, which benefits linear models by improving the model's performance on skewed data. The reason behind applying Yeo-Johnson here, as opposed to simpler transformations like logarithmic scaling, is that it handles both positive and negative values while maintaining the continuity of the data.

**Model Comparison and Selection**

For model training, we implemented three models:

1. **Lasso Regression**: A linear regression model with **L1 regularization**, designed to shrink coefficients of less important features to zero. This leads to a sparse model that performs well on small datasets like ours, where overfitting is a concern. Lasso tends to work well when only a subset of features contributes significantly to the output.
2. **ElasticNet**: This model combines both **L1 (Lasso)** and **L2 (Ridge)** regularization, balancing between shrinking irrelevant coefficients and penalizing large coefficients. This balance

helps improve the stability and performance of the model when the data is sparse or highly collinear.

3. **Random Forest Regressor**: Unlike the linear models above, **Random Forest** is an ensemble method that builds multiple decision trees and averages their predictions. While powerful and flexible, Random Forest can be prone to **overfitting**, especially on small datasets. It is also less interpretable compared to Lasso and ElasticNet.

Each model was tuned using **Grid Search CV** with **K-Fold Cross-Validation** to ensure that we were selecting the best hyperparameters based on the training data. This method helps guard against overfitting by testing different subsets of the data, giving a more reliable measure of model performance. Using **sample weights** during model training also ensured that the model learned effectively from the adjusted data, improving generalization.

**Complementary Use of Models and Weighted Averaging**

In the final step, we combined the three models using **Weighted Averaging**. This method assigns weights to the predictions from each model, with **Lasso** receiving the highest weight (0.5), **ElasticNet** at 0.3, and **Random Forest** at 0.2. The rationale behind this weighting lies in **Occam's Razor**—a principle that favors simpler models over more complex ones when both offer similar performance. While Random Forest performed well on training data, it risked overfitting due to the limited size of the dataset, which is why it received the lowest weight. On the other hand, **Lasso** was the most robust, offering stable performance without overfitting, which is why it was given the highest weight in the final predictions.

**Why Simplicity Wins: Occam's Razor**

In line with **Occam's Razor**, our final model intentionally leans towards simpler, more interpretable models. Despite the temptation to use more complex models like **Random Forest** or **CatBoost**, we found that these models overfit the training data. For instance, Random Forest achieved an RMSE of less than 1.0 on the training data but performed poorly on the test set, confirming overfitting. Conversely, the simpler models like Lasso and ElasticNet generalized better to unseen data, providing an RMSE of **1.8 on the training data** and **1.2 on the test data**.

In conclusion, the choice of a simpler linear model, combined with thoughtful feature engineering and weight adjustments, allowed us to achieve robust performance without overfitting.