MSc in Applied Data Science & Analytics

Data Science Applications Project Report

Proactive Analytics for Student Success: A Data-Driven Approach to Academic Outcomes

Submitted by: Mariem Nsiri
Supervisor: First Last

May 18, 2025

Declaration

I herby certify that this material, which I now submit for assessment on the programme of study leading to the award of MSc in Computing in the Institute of Technology Blanchardstown, is entirely my own work except where otherwise stated, and has not been submitted for assessment for an academic purpose at this or any other academic institution other than in partial fullfilment of the requirements of that stated above.

Author: Mariem Nsiri Dated: May 18, 2025

Data exploration techniques can be seen in the following pages:	pages 5 to 15
Data preparation techniques are described in the following pages:	pages 17 to 25

Abstract

Early identification of students at risk of poor academic performance is essential for timely intervention and improved educational outcomes. This project develops a predictive system leveraging data mining techniques applied to diverse student-related data, including behavioral, lifestyle, and psychological factors, to flag at-risk students weeks before final grades are released. Following the CRISP-DM methodology, the study begins with a clear definition of goals and constraints, followed by a thorough exploration and preparation of data. Advanced modeling techniques are then employed to capture complex relationships and accurately predict academic risk. The model's evaluation highlights key insights and limitations, demonstrating the practical potential of data-driven approaches in educational settings.

Contents

List of Tables

List of Figures

Chapter 1

Introduction

In today's educational landscape, early identification of students at risk of poor academic outcomes is crucial for providing timely support and improving overall success rates. Many institutions face challenges in detecting struggling students early enough to intervene effectively. This project addresses this need by leveraging data mining techniques to analyze diverse student-related data, including behavioral patterns, lifestyle choices, and psychological wellbeing, to build a predictive system that can flag at-risk students weeks before final grades are released.

This study begins by establishing a clear business understanding of the goals and constraints associated with early academic risk detection, as outlined in Chapter 2. Chapter 3 then examines the available data sources and assesses their quality. Next, comprehensive data preparation steps are carried out to ready the dataset for modeling, as detailed in Chapter 4.

The modeling phase involves selecting and training algorithms designed to capture complex relationships between variables and accurately predict student risk; this process is described in Chapter 5. Model evaluation, highlighting key findings, and limitations, is presented in Chapter 6. Finally, Chapter 7 draws conclusions and proposes future directions.

This structured workflow adheres to the CRISP-DM methodology, ensuring both analytical rigor and practical impact.

1. INTRODUCTION

Chapter 2

Business Understanding

Higher education institutions face increasing pressure to enhance student success and retention amid growing competition and limited resources. They must support a diverse student body with varied academic and personal needs. Although vast amounts of data are collected, universities often struggle to use it effectively due to fragmented and outdated systems that react only after issues arise. Consequently, student support tends to be reactive, with limited coordination among academic, engagement, and well-being services.

The rise of digital technology and social media has also transformed student engagement, introducing distractions and multitasking that reduce focus and motivation significantly. Many students feel overwhelmed by numerous future options, leading to confusion and lack of direction. Traditional support systems are ill-equipped to handle these challenges, highlighting the need for a proactive, data-driven approach. This approach should leverage predictive analytics and integrated data to identify at-risk students early and deliver timely, personalized support, as outlined in Sections 2.1 and 2.2.

2.1 Business objectives

The primary goal of this project is to build an integrated analytics system that gives a comprehensive understanding of the different factors influencing student performance. This system will help both students and staff take action early, supporting academic success through timely and personalized interventions. The main objectives are to:

- Develop a predictive model to detect at-risk students 6–8 weeks earlier than current methods, using academic, engagement, and behavioral data;
- Analyse non-academic drivers (e.g., digital habits, mental health, financial stress) affecting performance and engagement;
- Provide tailored support based on student profiles and learning preferences.

Success will be measured both quantitatively and qualitatively, though only selected metrics can be assessed within the scope of this project. The focus will be on achieving at least 80% accuracy in identifying at-risk students using historical data and evaluating how understandable the model outputs are for academic advisors. Broader outcomes such as improved retention or exam performance are acknowledged as longer-term goals. These could be evaluated in future studies through ongoing student and advisor feedback, as well as support from university leadership, which would indicate readiness for long-term use and wider institutional adoption.

Laying the groundwork This project relies on a synthetic dataset from Kaggle [Nath, 2025], which links student habits, such as study time, sleep, social media use, and mental health, to academic performance. Constraints include limited access to institutional data, a defined project timeline, and dependence on publicly available resources. The aim is to design an interpretable system that helps identify at-risk students early using academic and behavioral indicators. While broader outcomes like improved retention or exam performance are longer-term goals, they remain beyond the measurable scope of this initial work. Academic research on predictive models, student retention, and digital behavior may be referenced later during the data understanding phase.

2.2 Data mining objectives

This section outlines the primary data mining goals, focusing on two key objectives: classification and association. These goals aim to uncover actionable insights that support timely interventions and enhance student success. By applying these data mining techniques, we seek to enable early identification of at-risk students and optimize support strategies for improved academic outcomes.

- **Early risk classification:** Develop a classification model to identify students at risk of academic failure (e.g., grade $\leq$ D) 6–8 weeks in advance. The model will use predictors such as study habits, stress indicators, environmental factors, and non-academic influences (e.g., social media usage) to generate early warnings on academic performance.
- **Intervention optimization:** Discover frequent co-occurrences between behavioral and academic attributes using association rule mining (e.g., students with high social media usage and lower mental health ratings often show reduced attendance rates).

These goals support key business priorities such as reducing student attrition, improving academic performance, and guiding resource allocation. Data mining success will be measured by model performance (e.g., high recall in identifying at-risk students) and insights' value for timely action. This report primarily focuses on the first objective: early risk classification.

With the business objectives defined, the data must next be examined for the predictive model. Understanding of this data is considered crucial for the identification of at-risk students and the implementation of effective interventions. In the following chapter, the dataset will be explored and prepared for analysis to achieve the goals outlined earlier.

Chapter 3

Data Understanding

In this chapter, the data used to build the predictive model for identifying at-risk students is examined. Understanding the data is crucial for selecting relevant features, recognizing patterns, and addressing biases. This phase involves exploring the dataset, identifying key performance variables, and preparing the data for modeling. Insights will guide the development of an effective, data-driven system for student success.

Table 3.1: Dataset overview: key features and statistical properties

Feature	**Type**	**Description**	**Stats / Values**
student_id	Cat.	Unique student identifier	1,000 unique values
age	Num.	Age of the student	Mean: 20.5, Min: 17, Max: 24, Std: 2.31
gender	Cat.	Gender identity	Female (481), Male, Other
study_hours_per_day	Num.	Average daily study hours	Mean: 3.55, Min: 0, Max: 8.3, Std: 1.47
netflix_hours	Num.	Daily Netflix or entertainment hours	Mean: 1.82, Min: 0, Max: 5.4, Std: 1.08
part_time_job	Cat.	Whether the student has a part-time job	No (785), Yes
attendance_percentage	Num.	Class attendance percentage	Mean: 84.13%, Min: 56%, Max: 100%, Std: 9.4
sleep_hours	Num.	Average sleep hours per day	Mean: 6.47, Min: 3.2, Max: 10, Std: 1.23
diet_quality	Cat.	Self-reported quality of diet	Fair (437), Good, Poor
exercise_frequency	Num.	Weekly exercise frequency	Mean: 3.04, Min: 0, Max: 6, Std: 2.03
parental_education_level	Cat.	Highest parental education level	High School (392), Master, Bachelor (91 missing)
internet_quality	Cat.	Self-rated quality of internet	Good (447), Average, Poor
mental_health_rating	Num.	Self-assessed mental health (1-10 scale)	Mean: 5.44, Min: 1, Max: 10, Std: 2.85
extracurricular_participation	Cat.	Participation in these activities	No (682), Yes
exam_score	Num.	Final exam score	Mean: 69.6, Min: 18.4, Max: 100, Std: 16.89

The dataset, sourced from [Nath, 2025], contains 1,000 student entries and 16 features, including study hours, sleep patterns, social media usage, mental health rating, and exam scores. It is well-suited for this project, as it simulates realistic student behaviors.

Table 3.1 summarizes the dataset, which includes numerical and categorical features suitable for CRISP-DM techniques. A key data quality issue is 91 missing values (9.1%) in parental education level, limiting socio-educational analysis. Patterns reveal links between study time (mean: 3.55 h), attendance (84.13%), and exam scores (69.6). Behavioral factors like social media (2.51 h) and Netflix use (1.82 h) suggest distraction risks. Sleep (6.47 h), mental health (5.44), internet access, and parental education highlight disparities in student support and readiness.

Section 3.1 presents exploratory analysis, examining data distributions and links between study habits, attendance, behavior, and well-being. Section 3.2 addresses data quality, focusing on missing parental education data and bias risks. Section 3.3 highlights findings on academic performance, digital disparities, and targeted support recommendations.

3.1 Exploration

Exploratory analysis was conducted using targeted queries, histograms, and summary statistics to examine patterns in student behavior, lifestyle, and academic engagement. This step revealed trends and guided group-based analysis.

3.1.1 Initial data queries

The data were segmented by gender, the primary categorical grouping, revealing consistent exam scores and attendance rates across all groups. Average exam scores hovered around 69–70, with a notable concentration in the 99–100 range (28 females, 23 males, 1 other). Attendance rates were similarly high, averaging 84.4% for females and 83.9% for males (see Figure 3.1).

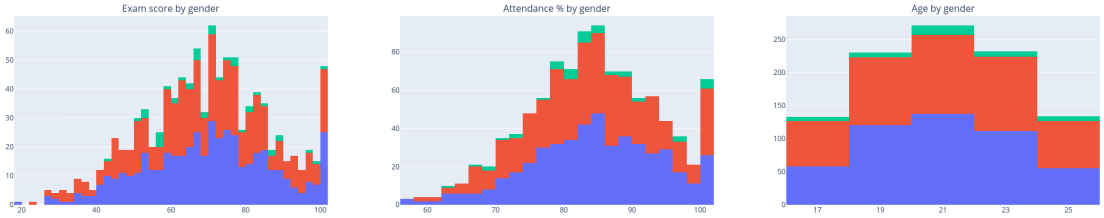

Figure 3.1: Distribution of exam scores, attendance, and age by gender

Further analysis was carried out by employment status. It was found that students with and without part-time jobs reported similar study duration (approximately 3.5 h/day), indicating that employment status did not significantly affect study time. Screen time metrics were also found to be consistent across gender groups, with slightly higher averages for students identifying as “Other” (2.59 h on social media and 2.06 h on Netflix) compared to approximately 2.5 and 1.8 h, respectively, for other groups. Sleep duration was observed to be stable across all categories, ranging from 6.4 to 6.8 h, with a marginally higher average among “Other” students (see Figure 3.2).

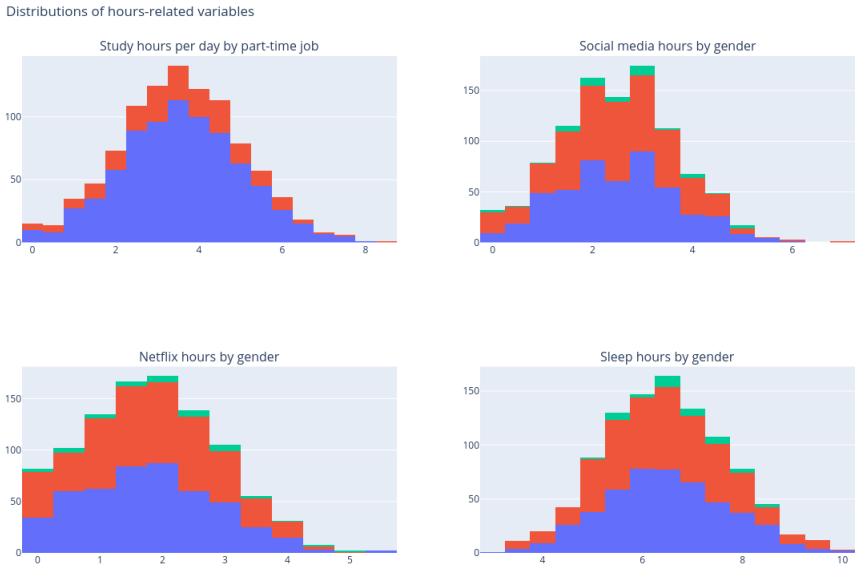

Figure 3.2: Distribution of study hours, screen time, and sleep

In summary, a relatively balanced routine was observed among students, with no substantial disparities in academic engagement or lifestyle habits based on gender or employment status.

Skewness of numerical data The skewness analysis (see code in Appendix A.1) of numerical attributes in the dataset suggests that most distributions are approximately symmetric or exhibit *mild skewness*. Attributes such as `study_hours_per_day` (0.0542), `social_media_hours` (0.1196), and `netflix_hours` (0.2368) show slight right skewness, indicating that a few students engage in these activities for a significantly higher amount of time. In contrast, `attendance_percentage` (-0.2375), `exercise_frequency` (-0.0319), and `exam_score` (-0.1561) exhibit slight left skewness, suggesting that most students tend to have higher attendance, more exercise, and better exam scores. Other attributes, such as `age` (0.0084), `sleep_hours` (0.0913), and `mental_health_rating` (0.0378), have skewness values close to zero, indicating approximate symmetry. Overall, none of skewness values exceed the typical threshold for problematic skewness (e.g., values greater than 1 or less than -1), meaning that **no data transformations**, such as logarithmic or Box-Cox transformations, **are necessary for modeling**, and the data is in a suitable state for analysis.

Distribution of categorical data

The dataset shows a relatively balanced gender distribution, with a smaller group identifying as "Other" . Most students do not hold part-time jobs, though slightly more males do than females. Diet quality is generally rated as fair, with poor diets slightly more frequent among females.

3. DATA UNDERSTANDING

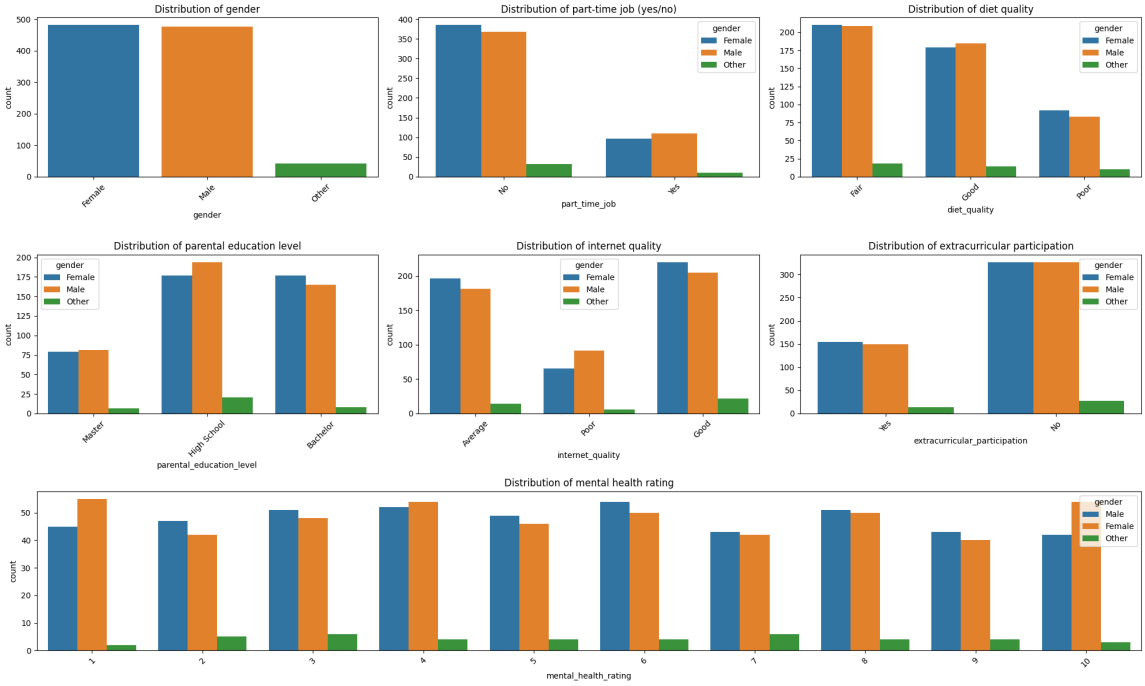

Figure 3.3: Distribution of categorical variables by gender

Parental education levels differ slightly by gender, with more males reporting parents with only a high school education, and more females reporting parents with bachelor's degrees. Internet quality is mostly rated as good or average, though poor connectivity appears slightly more among males. Participation in extracurricular activities is generally low, with similar rates between genders. Mental health ratings are broadly similar across genders, with most responses clustered around mid-range values (4 to 6). However, both ends of the scale are represented, indicating a range of experiences from very poor to excellent self-perceived mental well-being (see Figure 3.3).

3.1.2 Visualizing correlations, and aggregating data for insights

In this section, techniques are applied to visualize and analyze the data, turning it into actionable insights. Key metrics are aggregated and visualized to reveal patterns and correlations that inform how lifestyle habits may affect student performance. This helps identify areas needing support and offers a clearer view of factors influencing academic outcomes.

The correlation matrix reveals several key relationships within the dataset. A strong positive correlation was found between study hours per day and exam scores (0.83), indicating that higher study time is associated with better performance. A moderate positive correlation (0.32) was also observed between mental health rating and exam score, suggesting a possible link between well-being and academic outcomes. In contrast, age shows almost no correlation with exam scores (-0.01) or other variables.

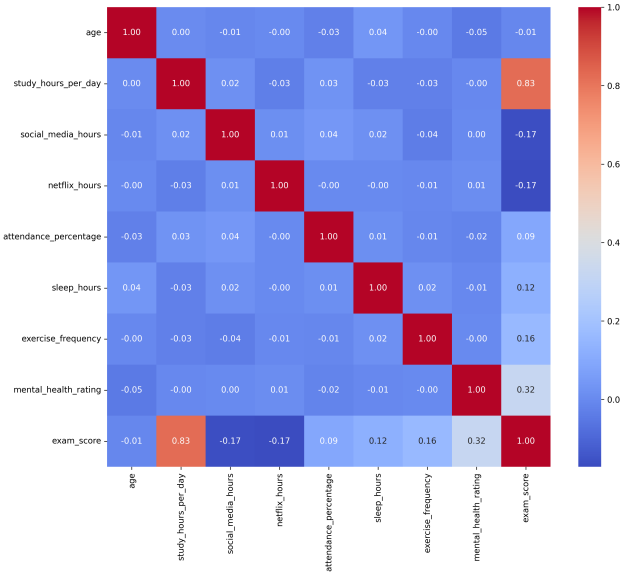

Figure 3.4: Correlation matrix (raw numerical data)

Weak negative correlations were identified between time spent on social media and Netflix with exam scores (-0.17), implying that increased screen time may be mildly detrimental to performance. Attendance percentage shows a weak positive correlation (0.09), while exercise frequency has a similarly weak association (0.16). Most other correlations in the dataset are minimal or negligible.

3.1.2.1 Numerical predictors

The scatter plot analysis confirms that study hours are the most reliable predictor of exam performance, with a consistent upward trend across all gender groups. Students generally gain between 9 and 11 exam points per additional hour of study, with R^2 values around 0.67 to 0.69, indicating strong predictive power. In contrast, sleep duration shows little to no consistent relationship with exam scores. The R^2 values remain below 0.03 for all groups, and while some trends are slightly positive or negative, none are practically significant. Similarly, attendance rates have negligible impact on performance, with flat trendlines and very low R^2 values, suggesting that presence alone is not a key driver of academic success in this dataset.

Media consumption shows modest but more noticeable patterns. Netflix hours correlate negatively with exam scores, especially among students identifying as "Other," where each additional hour corresponds to a decline of over 5 points and an R^2 of 0.238. Social media use exhibits a similar negative trend, with slightly higher R^2 values than sleep or attendance, particularly among female students. When exploring relationships between study time and lifestyle factors such as sleep or media use, no strong associations are found. The R^2 values are very low, and trendline slopes are near zero, indicating they explain virtually no variance in study time. This suggests that media habits and sleep do not meaningfully predict how much students study per day.

Figure 3.5: Relevant scatter plots.

Overall, lifestyle factors like sleep and media consumption appear to have limited influence on either exam scores or study duration when considered in isolation. More substantial factors, such as motivation, discipline, or external obligations, may better explain variations in academic behavior and outcomes.

3.1.2.2 Ordinal predictors

To assess associations involving ordinal predictors, two nonparametric rank-based correlation methods were employed. Spearman's ρ measures the strength of monotonic relations based on ranks, while Kendall's τ evaluates concordance between paired observations. These methods were selected due to the ordinal nature of several variables, rendering parametric approaches inappropriate.

Weak associations were observed between exam scores and the ordinal attributes of diet quality, parental education, and internet quality. Both Spearman's ρ and Kendall's τ were close to zero with non-significant p-values ($p > 0.05$), indicating no meaningful monotonic trends. Median exam scores showed minor fluctuations: highest for students reporting a "Fair" diet (71.0), slightly higher for those with parents holding a Bachelor's or High School degree, and unexpectedly higher for those with poor internet quality (71.4). These modest variations and low correlations suggest that these factors do not substantially influence performance in this dataset (see code Appendix A.2).

As mental health rating was treated as an ordinal variable, scatter plots were not used. Instead, correlation analysis revealed a statistically significant positive association with exam performance. Spearman's correlation coefficient was $\rho = 0.323$ ($p < 0.001$), and Kendall's $\tau = 0.228$ ($p < 0.001$), both indicating a moderate monotonic relationship. Exam score medians increased steadily from 62.75 at the lowest mental health rating (1) to 79.90 at the highest (10), with wide interquartile ranges, suggesting that students with better self-reported mental health tend to achieve higher scores, despite individual variability.

In contrast, negligible correlations were found between mental health rating and media consumption. For Netflix viewing hours, $\rho = 0.003$ and $\tau = 0.001$; for social media hours, $\rho = -0.004$ and $\tau = -0.003$, with all p-values exceeding 0.89. These results indicate no significant associations between mental health and time spent on these platforms. A group-by summary supports this: median Netflix hours ranged narrowly (1.50 – 1.95) with stable IQRs (InterQuartile Ranges), while social media use showed slightly broader medians (2.40 – 2.80) and IQRs (1.40 – 1.85). These small differences suggest media use is not a reliable indicator of mental health in this sample.

3.1.3 Outlier analysis

To understand variability in student behaviors and outcomes, an outlier analysis was conducted on key continuous variables. For study hours per day, most students reported studying between 2.6 and 4.5 hours, with a few notable outliers reaching up to 8.3 hours. Similar patterns were found in social media hours (typical range: 1.7–3.3 hours; outliers: 6.2 and 7.2) and Netflix hours (most between 1.0–2.5 hours; outliers above 5 hours), suggesting a small number of individuals deviate substantially from the norm (see Figure 3.6, top).

Figure 3.6: Box plots of exam scores by numerical variables (top) and categorical variables, including outliers (bottom).

Attendance percentage was generally high (78%–91%), though some students had much lower attendance (as low as 56%), indicating disengagement. Sleep hours were fairly consistent (5.6–7.3 hours), with occasional long sleepers reaching 10 hours. In contrast, exam score exhibited the widest spread (18.4–100), with most scores falling between 58.5 and 81.3. Notably, low-end outliers (e.g., 18.4 and 23.1) point to significant academic underperformance.

These findings suggest that while most behaviors cluster tightly, variables like exam score and attendance reveal broader disparities, with extreme values potentially highlighting students in need of support or intervention.

3. DATA UNDERSTANDING

Exam scores show modest variation across categorical attributes (see Figure 3.6, bottom). Students without part-time jobs slightly outperform those with jobs (69.84 vs. 68.74), and those reporting a fair diet have the highest average scores (70.43), followed by good (69.37) and poor (68.13) diets. Parental education shows mixed trends: students with a bachelor's degree score highest (70.27), while those with a master's score lower (68.09); unknown education levels still yield relatively high scores (70.03). Interestingly, students with average internet quality perform best (70.64), exceeding both poor (69.72) and good (68.65) connections. Extracurricular activities shows no substantial difference, with nearly identical averages for participants (69.62) and non-participants (69.59).

Across all categories, a few low-end outliers are present, indicating that some students perform considerably worse than their peers regardless of background. These may reflect individual challenges unrelated to the measured attributes, such as personal, health, or motivational factors.

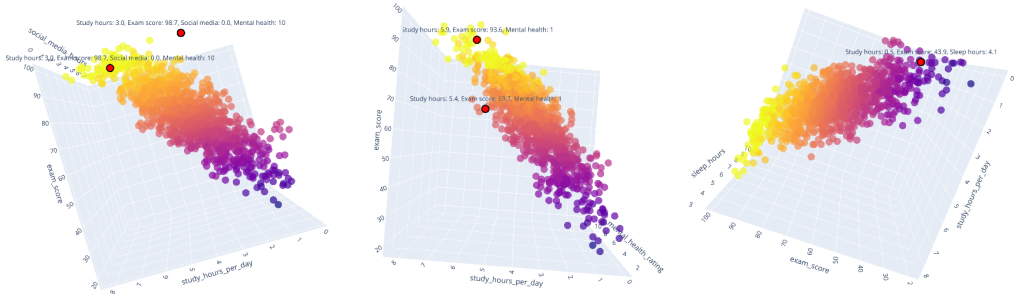

Figure 3.7: Relevant multivariate outliers: (Left) High exam score despite minimal study hours, (Middle) High academic performance with low mental health, (Right) Low study hours and sleep with decent exam score.

Analysis of relevant multivariate outliers Some Multivariate outliers were identified manually through interactive 3D scatter plots. One notable case, shown in Figure 3.7 (right), involves a student (row 712) who achieved a high exam score of 98.7 despite studying only 3.0 hours per day. This student also reported perfect mental health (10) and no time spent on social media. Such a profile suggests the presence of highly efficient learners whose strong cognitive ability, psychological well-being, and limited distractions may compensate for shorter study durations.

Two additional outliers (rows 447 and 998), shown in Figure 3.7 (middle), scored well on exams (93.6 and 69.7) but reported very low mental health rating (1). Despite their academic success, these students show signs of stress, highlighting the need for mental health support even among high performers. This emphasizes the importance of considering mental well-being alongside academic achievement. Another outlier, row 92, shown in Figure 3.7 (left), presents an unusual case where a student reported only 0.5 hours of daily study but achieved a respectable exam score of 43.9, despite getting only 4.1 hours of sleep. This suggests that the link between study time and exam performance may not always be linear, with other factors, such as individual learning efficiency or external support, potentially influencing outcomes.

Overall, the identified outliers likely reflect natural variations in behavior and can generally be retained. High values in study hours, social media, Netflix use, and sleep likely stem from individual lifestyle choices. Outliers in attendance and exam scores, particularly at the lower end, may signal disengagement or academic struggles and could benefit from further attention.

3.1.4 Statistical analysis

Several statistical tests were conducted to evaluate the differences in exam scores across different groups. First, the null hypothesis was tested to determine if there was a significant difference in exam scores between males and females. The T-statistic of -0.339 and the P-value of 0.735 were obtained, leading to the failure to reject the null hypothesis, indicating that no significant difference in exam scores between the two groups was found. Similarly, when comparing Male/Female students to Other students, a T-statistic of -0.410 and a P-value of 0.682 were observed, resulting in the failure to reject the null hypothesis, showing that no significant difference was found between these two groups.

Next, binning was applied to categorize data into groups for One-Way ANOVA to assess the effect of mental health ratings, social media usage, Netflix usage, and media usage categories on exam scores. Specifically, social media hours and Netflix hours were binned into three categories: Low, Moderate, and High, based on the 25th and 75th percentiles. Significant differences in exam scores across these factors were identified. For mental health ratings, an F-statistic of 51.22 and a P-value of 6.67×10^{-22} were obtained, indicating a significant difference between the groups. Both social media and Netflix usage levels were found to have significant effects on exam scores, with P-values of 2.78×10^{-5} and 3.28×10^{-6}, respectively.

```
# Combine social media hours and Netflix hours into a new variable 'total_media_usage'
sd['total_media_usage'] = sd['social_media_hours'] + sd['netflix_hours']
# Create the 'media_usage' categorical variable based on distribution
# Using the summary stats from the provided data for setting thresholds
low_threshold = sd['total_media_usage'].quantile(0.25) # 25th percentile
high_threshold = sd['total_media_usage'].quantile(0.75) # 75th percentile
# Define the 'media_usage' category
sd['media_usage'] = pd.cut(sd['total_media_usage'],
                           bins=[-float('inf'), low_threshold, high_threshold, float('inf')],
                           labels=['Low', 'Moderate', 'High'])
# Perform One-Way ANOVA to test if exam scores differ across the 'media_usage' categories
# Group exam scores by 'media_usage'
low_media_scores = sd[sd['media_usage'] == 'Low']['exam_score']
moderate_media_scores = sd[sd['media_usage'] == 'Moderate']['exam_score']
high_media_scores = sd[sd['media_usage'] == 'High']['exam_score']
# ... see code 2 in Appendix A.3
```

Finally, a significant difference in exam scores across media usage categories (Low, Moderate, High) was found, with an F-statistic of 26.17 and a P-value of 8.33×10^{-12}. Consequently, the null hypothesis was rejected for all these factors, and it was concluded that media usage significantly affects exam scores (see code 2 in Appendix A.3).

After exploring the dataset and its patterns, we now focus on its quality. The dataset shows no significant issues, and further analysis is unnecessary, as it is reliable for subsequent use.

3.2 Data quality

This is the final section of the data understanding phase. Given the overall good quality of the dataset, there is no need for an extensive analysis of data quality. A brief check for outliers using the Z-score method was sufficient to confirm that no significant issues exist.

```
from scipy.stats import zscore
num_cols = sd.select_dtypes(include=np.number).columns
z_scores = np.abs(zscore(sd[num_cols]))
outliers = (z_scores > 3)
print(f"Deviations detected per column: {outliers.sum()}")
```

```
Deviations detected per column: 11
```

The Z-score analysis identified 11 deviations across numerical columns, but these are likely legitimate extremes (e.g., high study hours or low sleep) rather than noise. As the dataset shows no major skew and all values are plausible, there is no need to remove them. These deviations should be retained as they reflect diverse student habits and may provide valuable insights into performance patterns.

```
# study_hours + netflix_hours + social_media_hours should not exceed 24.
sd['total_activity_hours'] = sd['study_hours_per_day'] + sd['social_media_hours'] + sd['netflix_hours']
print("Entries with total activity > 24:", (sd['total_activity_hours'] > 24).sum())
```

```
Entries with total activity > 24: 0
```

Since the output is 0, it confirms that no student exceeds 24 total activity hours per day when combining study, social media, and Netflix time. This suggests that the data is internally consistent in terms of time allocation and does not contain unrealistic values in these features. It's a good sign that the dataset doesn't exhibit this form of noise or fabrication.

```
# Logical anomaly check: High exam score with low study hours
anomaly_df = sd[(sd['study_hours_per_day'] < 1) & (sd['exam_score'] > 85)]
print("Possible anomalies (High score with very low study time): ", anomaly_df[['study_hours_per_day', 'exam_score']])
```

```
Possible anomalies (High score with very low study time):  Empty DataFrame
Columns: [study_hours_per_day, exam_score]
Index: []
```

The fact that the query returned an empty DataFrame means there are no students who scored above 85 with less than 1 hour of study per day. This absence of logical anomalies supports the credibility and realism of the dataset,there are no implausible cases of extremely high performance without corresponding effort, at least by the study hours metric. This adds further confidence that the data is free from major logical noise or inconsistencies.

```
# Logical anomaly: Very low sleep but high attendance
sleep_attendance_check = sd[(sd['sleep_hours'] < 3) & (sd['attendance_percentage'] > 90)]
print("Unusual sleep-attendance combinations: ", sleep_attendance_check[['sleep_hours', 'attendance_percentage']])
```

```
Unusual sleep-attendance combinations:  Empty DataFrame
Columns: [sleep_hours, attendance_percentage]
Index: []
```

The empty result confirms that there are no unrealistic cases of students sleeping less than 3 hours regularly while maintaining attendance above 90%. This further strengthens the internal consistency of the dataset and indicates that it doesn't contain logically contradictory or biologically implausible records. These kinds of checks help validate that the data is not only statistically clean but also realistically grounded in expected human behavior.

3.3 Final findings

The dataset was thoroughly reviewed for completeness, consistency, and validity. It is largely complete, with only a small gap in the variable related to parental education level. There are no duplicates, and mild outliers in key attributes remain within acceptable limits. Skewness in study-related habits, media usage, and entertainment time is minimal, and no transformations are necessary. Logical validation confirmed the absence of extreme or contradictory behavior, ensuring the dataset's consistency and suitability for modeling.

Key insights from the analysis showed that academic success is most strongly influenced by study hours and mental health ratings. Media usage has a small but significant negative effect on performance, while attendance, gender, and part-time employment showed minimal impact. Statistical tests further confirmed that both media usage and mental health significantly affect exam scores, underlining their importance for predictive modeling.

These findings align seamlessly with the business objectives (see Section 2.1), ensuring that the data supports the development of effective interventions aimed at improving student performance. With these insights in mind, we can now move forward to the data preparation phase, where we will ready the dataset for model implementation.

Chapter 4

Data Preparation

Building on the foundation established in Chapter 2 and the insights from Chapter 3, this phase of the CRISP-DM process prepares the dataset for modeling. This chapter outlines the selection of key attributes and records, necessary transformations, and final data cleaning steps. The aim is to create a clean, efficient dataset that aligns with business and analytical goals.

As discussed in Sections 3.1 to 3.3, the dataset comprises 16 attributes and around 1,000 records, with minimal quality issues. No sampling or reduction is needed. A binary target label is created from the exam score in Section 4.1, followed by baseline evaluation in Section 4.2, attribute selection in Section 4.5.6, data cleaning in Section 4.4, feature construction in Section 4.5, and final pipeline validation in Section 4.6.

4.1 Generate a binary classification label

To establish a classification target, it was assumed that `exam_score` represents an aggregate or average across all academic modules. Based on this, a new categorical variable, `grade`, was created by mapping numerical scores to standard letter grades from A to F, excluding subcategories such as A− or B+. This allowed for broader performance groupings. A binary label, `risk_status`, was then derived to indicate whether a student is *AtRisk* (grade D or F) or *NotAtRisk* (above D), thereby framing the problem as a binary classification task.

```
sd = pd.read_csv("data/student_habits_performance.csv")
def convert_to_letter_grade(score):
   return 'A' if score >= 80 else 'B' if score >= 70 else 'C' if score >= 60 else 'D' if score >= 50 else 'F'
# Create 'grade' attribute
sd['grade'] = sd['exam_score'].apply(convert_to_letter_grade)
# print(sd['grade'].unique())
grade_counts = sd['grade'].value_counts().sort_index()
print(grade_counts)
```

```
grade
A    277
B    234
C    209
D    149
F    131
```

The grade distribution shows that most students are performing well, with `A` being the most common grade (277 students). `B` (234 students) and `C` (209 students) are also prevalent, while fewer students have `D` (149 students) and `F` (131 students) grades. This indicates a mix of strong and weaker performers, highlighting the need for interventions for students in the `D` and `F` categories to improve academic success. The code below creates a new column `risk_status`. It checks each student's `grade` and assigns `'AtRisk'` if the grade is `D` or `F`, and `'NotAtRisk'` for all other grades.

```
sd['risk_status'] = sd['grade'].apply(lambda x: 'AtRisk' if x in ['D', 'F'] else 'NotAtRisk')
```

```
risk_status
AtRisk       280
NotAtRisk    720
```

This output shows a class imbalance in the `risk_status` variable. There are 720 students labeled as `NotAtRisk` and only 280 as `AtRisk`. This imbalance—approximately 72% vs. 28%—should be taken into account when building the classification model, as it may bias the model toward predicting the majority class.

4.2 Baseline model

Before starting the preprocessing phase, it is useful to build simple baseline models to set a reference point for performance. These models help evaluate whether future improvements, such as feature selection or data transformation, actually lead to better results. A baseline model is typically created using the raw or minimally processed data and serves as a benchmark for comparison throughout the analysis.

```
sd_y = sd['risk_status']
sd_X = sd[sd.columns.difference(['risk_status'])]

from sklearn.preprocessing import OrdinalEncoder
sd_XX = sd_X.copy()
num_cols = list(sd_XX.select_dtypes(include=['int64','float64']).columns)
cat_cols = list(sd_XX.select_dtypes(include=['object']).columns)
# sd_XX[num_cols] = sd_XX[num_cols].apply(lambda col: col.fillna(0, axis=0)) # Non applicable
sd_XX[cat_cols] = sd_XX[cat_cols].apply(lambda col: col.fillna('?', axis=0).astype(str))
ordinal_encoder = OrdinalEncoder()
sd_XX[cat_cols] = ordinal_encoder.fit_transform(sd_XX[cat_cols])
```

```
from sklearn.tree import DecisionTreeClassifier, plot_tree, export_graphviz
import adsa_utils as ad
# Create an instance of the DT classifier
clf = DecisionTreeClassifier(random_state=43)
# Crossvalidate on the encoded train portion
ad.cross_validation_avg_scores(sd_XX, sd_y, clf)
```

```
Mean Accuracy is 100.00 +/- 0.00
Mean Precision is 100.00 +/- 0.00
Mean Recall is 100.00 +/- 0.00
Mean F-score is 100.00 +/- 0.0
```

The perfect performance observed should be questioned, prompting a deeper exploration of the dataset. It is important to review the feature definitions or investigate the tree model to better understand the results.

```
sd_XX = sd_XX.drop(columns=['exam_score', 'grade'])
ad.cross_validation_avg_scores(sd_XX, sd_y, clf)
```

```
Mean Accuracy is 84.10 +/- 1.32
Mean Precision is 80.32 +/- 1.65
Mean Recall is 80.12 +/- 1.69
Mean F-score is 80.22 +/- 1.66
```

The decrease in performance after removing the *proxy* attributes `exam_score` and `grade` highlights their strong predictive influence on the target variable. This underscores the importance of selecting features that are both informative and realistically available at the time of prediction, particularly in early intervention contexts.

```
sd_XX = sd_XX.drop(columns=['exam_score', 'grade'])
ad.cross_validation_avg_scores(sd_XX, sd_y, clf)
```

4.3 Data selection

The first step in preparing the data involves selecting the most relevant columns and rows to align with the objectives of the analysis. These decisions were guided by insights from the data understanding phase. Features were retained based on their demonstrated relationship with overall exam performance. At this stage, the entire dataset is preserved, except for attributes that do not contribute to analysis. For example, `student_id`, being a unique identifier, is removed as it offers no analytical value for modeling or interpretation. These results confirm that removing `student_id` has no negative impact on model performance, as expected.

```
sd_XX.drop(columns='student_id', inplace=True)
ad.cross_validation_avg_scores(sd_XX, sd_y, clf)
```

```
Mean Accuracy is 85.00 +/- 2.17
Mean Precision is 81.45 +/- 2.71
Mean Recall is 81.51 +/- 2.33
Mean F-score is 81.45 +/- 2.50
```

Based on the findings of Chapter 3 and summary statistics, all attributes, except for the already removed ID column, exhibit meaningful variability and are suitable for analysis. None show constant or near-constant values, and most have a reasonable spread, supporting their potential usefulness in identifying patterns related to academic performance. To confirm the absence of redundancy, it is also worthwhile to test feature correlation using **SmartCorrelatedSelection**.

```
from feature_engine.selection import SmartCorrelatedSelection
# Configure the SmartCorrelationSelection feature engine
scs_selector = SmartCorrelatedSelection(
                variables=None,
                method="pearson",
                threshold=0.95,
                selection_method="variance",
                estimator=None)
sd_XX = scs_selector.fit_transform(sd_XX)
print(f"Highly correlated feature sets (potential multicollinearity): {scs_selector.correlated_feature_sets_}")
print(f"Features recommended for removal due to high correlation: {scs_selector.features_to_drop_}")
```

```
Highly correlated feature sets (potential multicollinearity): []
Features recommended for removal due to high correlation: []
```

The results show that there are no highly correlated feature sets and no features recommended for removal based on the specified correlation threshold. This confirms that all features (excluding the ID) are sufficiently distinct and should be retained for analysis.

4.4 Preparing the groundwork: data cleaning

This phase addresses the remaining quality issues by confirming outliers and handling missing values, ensuring the dataset is fully prepared for modeling. The attribute `parental_education_level` is the only feature with missing values. Given its moderate importance as a socioeconomic indicator, the missing entries are imputed using the mode. This method preserves the original distribution and aligns well with the categorical, ordinal nature of the variable.

Two imputation methods were evaluated for missing values in the parental education attribute: KNN imputation and mode imputation. The KNN-based approach yielded a mean accuracy of 84.70% and F1-score of 80.99%, while mode imputation produced slightly better results, with an accuracy of 85.80% and F1-score of 82.04%. Given its simplicity, lower computational cost, and stronger performance, mode imputation was preferred (see Appendix A.4). Since class-based mode imputation gave identical results to global mode for `parental_education_level`, the simpler strategy was adopted. It also preserves categorical structure without adding unnecessary complexity.

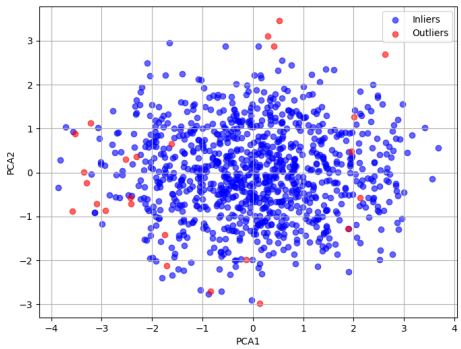

Figure 4.1: Multivariate outlier detection using Isolation Forest with PCA projection.

The outliers identified in the dataset, representing both extremely low- and high-performing students, offer valuable insights into the student population and should not be removed, as they reflect real-world extremes that may enhance model performance. The Isolation Forest algorithm, configured with `contamination=0.025` and `random_state=42`, effectively detected subtle anomalies without flagging valid data, striking a balance between sensitivity and precision (see Appendix A.5). Several outliers previously observed during the data understanding phase (e.g., rows 712, 447, 998, and 92) were also detected by the Isolation Forest model, reinforcing the validity of those earlier findings (see Section 3.1.3).

To better visualize the multivariate outliers detected by the Isolation Forest algorithm, Principal Component Analysis (PCA) [Pearson, 1901] was applied to reduce the high-dimensional feature space to two principal components. This transformation preserves as much variance as possible while enabling clear 2D plotting of complex relationships among variables. The resulting PCA projection highlights anomalous data points that deviate significantly from the general distribution, aiding interpretation and validation of the unsupervised outlier detection (see Figure 4.1).

To minimize the potential impact of these outliers on predictive performance, the Decision Tree Descritser technique, which is inherently robust to outliers, will be employed in the subsequent preprocessing steps. This approach supports reliable feature transformation while preserving the integrity of the original data.

4.5 Feature construction and preprocessing

In this phase, a transition is made from full-dataset transformations to a modelling-aware pipeline where data preprocessing is applied only to the *training set*. This approach is adopted to prevent data leakage and to yield more realistic performance estimates. At this stage, a Decision Tree Classifier is used as a baseline model to assess the impact of each preprocessing step. It was selected for its simplicity, interpretability, and compatibility with later feature selection.

A stratified 80/20 split was performed to preserve the class distribution across training and test sets. Stratification was considered essential due to the class imbalance between the two classes `AtRisk` and `NotAtRisk` students.

4.5.1 Deriving media usage with PCA

Principal Component Analysis (PCA) [Pearson, 1901] is a statistical technique used to transform correlated variables into uncorrelated components that capture the main patterns in the data. PCA was applied to the highly correlated features `social_media_hours` and `netflix_hours` to construct a new attribute, `media_usage_hours_pca`.

```
sd_XX = sd_X.copy()
pca = PCA(n_components=1)
sd_XX['media_usage_hours_pca'] = pca.fit_transform(sd_XX[['social_media_hours', 'netflix_hours']])
# Drop ID and proxy attributes
sd_XX.drop(columns=['student_id', 'exam_score', 'grade'], inplace=True)
```

This derived variable captures the shared variance between the two original inputs and summarizes overall media consumption in a single, informative component. Importantly, the original features were retained, and this new component was added to enrich the feature space without discarding existing information.

4.5.2 Supervised discretization with Decision Tree

Continuous features were discretized using the **DecisionTreeDiscretiser** [Fayyad and Irani, 1993], which applies supervised binning based on class labels. Unlike unsupervised methods (e.g., equal-width or quantile binning), this approach captures non-linear patterns by creating class-informed intervals, improving robustness to noise and preserving predictive structure.

```
# Apply discretization
dt_binner = DecisionTreeDiscretiser(
     bin_output='boundaries',
     cv=5,
     precision=2,
     scoring='accuracy',
     param_grid={'max_depth': [3, 4, 5, 6], 'ccp_alpha': [0.0, 0.005, 0.01]}
)
X_train_binned = dt_binner.fit_transform(X_train, y_train)
X_test_binned = dt_binner.transform(X_test)
X_train = pd.DataFrame(X_train_binned)
X_test = pd.DataFrame(X_test_binned)
y_train = pd.Series(y_train)
```

To improve the stability of the splits, cross-validation was used to tune the tree depth and pruning parameters. The resulting transformation consistently led to improved downstream model performance, particularly in cases where complex or non-linear feature-target relationships were present, and traditional binning failed to capture such interactions.

4.5.3 Categorical encoding and imputation

Categorical variables were processed using a two-step pipeline composed of an imputation and encoding stage:

- **Simple Imputer** was applied to replace missing values with the most frequent category. This strategy is equivalent to the mode imputation method previously discussed for the `parental_education_level` attribute (see Section 4.4).
- **Ordinal Encoder** was used to convert categorical values into numeric form. As no parameters for unknown category handling were specified, the encoder assumes all category levels are known during training and inference.

```
# Pipeline for categorical columns: mode imputation + ordinal encoding
cat_cols = X_train.select_dtypes(include=['object']).columns.tolist()
preprocessor = Pipeline([
   ('imputer', SimpleImputer(strategy='most_frequent')),
   ('encoder', OrdinalEncoder()
])
X_train = pd.DataFrame(preprocessor.fit_transform(X_train[cat_cols]), columns=cat_cols)
X_test = pd.DataFrame(preprocessor.transform(X_test[cat_cols]), columns=cat_cols)
```

This approach was selected for its compatibility with tree-based models, which can leverage the ordinal nature of encoded categories without requiring one-hot encoding. Additionally, the combined imputation and encoding process ensured that both missing and rare values were handled systematically, improving robustness and generalization.

4.5.4 Class imbalance handling: Borderline SMOTE

Class imbalance was a key concern identified during exploration, particularly between the `AtRisk` and `NotAtRisk` student categories. Such imbalance can bias learning algorithms toward the majority class, leading to poor generalization on the minority class. To mitigate this, the training data was augmented using **BorderlineSMOTE** [Han et al., 2005], an advanced oversampling technique that generates synthetic samples near the decision boundary:

```
smote = BorderlineSMOTE(random_state=43, k_neighbors=5, m_neighbors=10)
X_train, y_train = smote.fit_resample(X_train, y_train)
```

Unlike standard SMOTE, which interpolates new instances across the entire feature space, BorderlineSMOTE focuses on "danger" regions where minority samples are at risk of being misclassified. This localized synthesis helps better define the class boundary, improving model sensitivity to the minority class. The parameters `k_neighbors` and `m_neighbors` control the neighborhood size used to identify informative borderline samples, enhancing the quality of the synthetic data.

While BorderlineSMOTE showed slightly lower accuracy (89.59% vs. 90.19%), it exhibited more stable performance with smaller standard deviations across metrics, indicating higher confidence in its results. The improved F1-score for the minority `AtRisk` class further supports its effectiveness, particularly in handling class boundaries.

4.5.5 Scaling

Standardization was applied to ensure features have zero mean and unit variance. While not strictly necessary for tree-based models, it was included to preserve pipeline modularity and in case of later extension to linear models:

```
scaler = StandardScaler()
X_train = pd.DataFrame(scaler.fit_transform(X_train), columns=cat_cols)
X_test = pd.DataFrame(scaler.transform(X_test), columns=cat_cols)
```

4.5.6 Embedded feature selection

To reduce the dimensionality of the feature set and enhance the model's generalization, **SelectFromModel** was applied in conjunction with a **DecisionTreeClassifier**. This embedded feature selection method works by evaluating feature importance, with less important features being discarded based on a predefined threshold. The **DecisionTreeClassifier** was tuned with hyperparameters such as a maximum depth of 7, a minimum samples split of 4, a minimum samples leaf of 10, and a complexity parameter `ccp_alpha` of 0.007 to control overfitting and improve the overall model's interpretability.

This method ranks the features according to their contribution to the decision-making process of the model. Features with importance scores below the median are removed, allowing the model to focus only on the most relevant features. This approach is both computationally efficient and interpretable, as it provides a clear rationale for which features were selected and discarded.

```
selector_estimator = DecisionTreeClassifier(max_depth=7, min_samples_split=4,
                                            min_samples_leaf=10, ccp_alpha=0.007,
                                            random_state=43)
sfm = SelectFromModel(estimator=selector_estimator, threshold='median')
X_train_selected = sfm.fit_transform(X_train, y_train)
X_test_selected = sfm.transform(X_test)
selected_features = [cat_cols[index] for index in sfm.get_support(indices=True)]
X_train = pd.DataFrame(X_train_selected, columns=selected_features)
X_test = pd.DataFrame(X_test_selected, columns=selected_features)
print(selected_features)
```

```
['age', 'attendance_percentage', 'diet_quality', 'exercise_frequency',
 'extracurricular_participation', 'gender', 'internet_quality',
 'mental_health_rating', 'netflix_hours', 'parental_education_level',
 'part_time_job', 'sleep_hours', 'social_media_hours', 'study_hours_per_day',
 'media_usage_hours_pca']
```

No features were removed from the initial set, except those deliberately excluded during prior data preprocessing. This process ensured that all initially retained attributes remained, as they were considered relevant for the model.

4.6 Preprocessing pipeline final validation

To evaluate the impact of preprocessing, hyperparameter optimization was performed using **RandomizedSearchCV** on a decision tree. The search explored 300 random combinations of hyperparameters:

```
params_dist = {
  'splitter': ['best'],
  'criterion': ['gini', 'entropy'],
  'max_depth': range(5, 7),
  'min_impurity_decrease': np.linspace(0.005, 0.05, 10),
  'min_samples_leaf': range(4, 16, 2),
  'min_samples_split': range(4, 12, 2),
  'ccp_alpha': uniform(loc=0.005, scale=0.004)
}
random_search.fit(X_train, y_train)
print(f'Best parameters found: {random_search.best_params_}')
```

```
Best parameters found: 'ccp_alpha': 0.006731623306267057, 'criterion': entropy', 'max_depth': 6, 'min_impurity_decrease': 0.005,
                       'min_samples_leaf': 6, 'min_samples_split': 8, 'splitter': 'best'
```

```
best_clf.get_depth(), best_clf.get_n_leaves()
```

```
(6, 19)
```

The hyperparameter tuning results show that the best-performing decision tree model has a depth of 6, 19 leaves, and utilizes the `entropy` criterion with moderate pruning (`ccp_alpha` = 0.0067) to balance overfitting. The model achieves a mean accuracy of 89.59% (±1.04%), with a mean precision of 89.87% (±0.94%) and recall of 89.59% (±1.05%), indicating strong overall performance.

While the model performs well across both classes, the precision for the `NotAtRisk` class (0.93) is slightly higher than for `AtRisk` (0.75), reflecting the class imbalance. With a recall of 0.84 for `AtRisk` and 0.89 for `NotAtRisk`, the model shows a solid ability to correctly identify both classes, and the F1-score of 89.57% (±1.05%) suggests a good balance between precision and recall (see Figure 4.2 and full code in Appendix A.6). The decision tree's relatively shallow depth and moderate leaf size help prevent overfitting while still capturing the necessary complexity in the data.

```
best_clf = random_search.best_estimator_
ad.cross_validation_avg_scores(X_train, y_train, best_clf, cv_=5, print_dict=False)
y_pred = best_clf.predict(X_test)
print(ad.classification_report(y_test, y_pred))
ad.plot_save_confusion_matrix(y_test, y_pred, "confusion_matrix_dt")
```

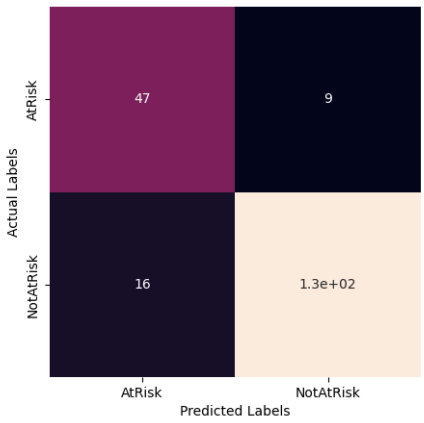

Figure 4.2: Confusion matrix of the best decision tree model after hyperparameter tuning.

```
Mean Accuracy is 89.59 +/- 1.04
Mean Precision is 89.87 +/- 0.94
Mean Recall is 89.59 +/- 1.05
Mean F-score is 89.57 +/- 1.05
              precision    recall  f1-score   support

      AtRisk       0.75      0.84      0.79        56
   NotAtRisk       0.93      0.89      0.91       144

    accuracy                           0.88       200
   macro avg       0.84      0.86      0.85       200
weighted avg       0.88      0.88      0.88       200
```

The next step is the modeling process, which will involve testing more powerful algorithms, such as Random Forest and XGBoost, to further improve performance. All preprocessing and feature selection steps are already in place, ensuring a smooth transition to these advanced models. Additionally, the embedded feature selection method will be updated to align with the corresponding algorithm, ensuring consistency and coherence throughout the modeling pipeline.

Chapter 5

Modeling

Building on the data preparation and preprocessing pipeline detailed in Chapter 4, this chapter focuses on developing and evaluating predictive models to accurately classify students at risk. A decision tree model provided a reliable baseline, confirming the effectiveness of earlier transformations and revealing key dataset relationships.

This chapter compares two advanced ensemble methods: **Random Forest** [Breiman, 2001] and **Extreme Gradient Boosting XGBoost** [Chen and Guestrin, 2016]. Both models excel at handling complex, imbalanced data while capturing non-linear patterns. Embedded feature selection techniques are applied in line with each model's structure. Hyperparameter tuning is conducted to optimize performance. Cross-validation ensures model robustness. Hold-out set evaluations assess generalization. These steps are detailed in Sections 5.1 and 5.2.

5.1 Ensemble-based classification of At-Risk students

The goal of this section is to build predictive models capable of accurately classifying students as `AtRisk` or `NotAtRisk`. Leveraging the refined features from the preprocessing pipeline, this stage applies ensemble methods to improve classification performance over the decision tree baseline. The analysis begins with the Random Forest model, followed by XGBoost, each evaluated through cross-validation and tested on a hold-out set. Performance metrics such as accuracy, precision, recall, and F1-score are used to assess model effectiveness, particularly with regard to correctly identifying students at risk.

5.1.1 Feature selection process

To ensure consistency in the feature set and optimize performance, the feature selection process was standardized using XGBoost's embedded feature importance method via the `SelectFromModel` approach. Although both Random Forest and XGBoost are capable of performing embedded feature selection, the attributes deemed important by each model can differ. Since XGBoost is

expected to outperform Random Forest due to its boosting mechanism and superior handling of complex interactions, its feature selection was used as a common basis for both classifiers. This strategy ensures that all models are trained on the same, optimally selected feature set, while maximizing the predictive potential of the XGBoost classifier.

```
# Convert target labels to numeric format (required for XGBoost classification)
label_encoder = LabelEncoder()
y_train_encoded = label_encoder.fit_transform(y_train)
y_test_encoded = label_encoder.transform(y_test)
## Make y_train and y_test pandas Series again
y_train = pd.Series(y_train_encoded)
y_test = pd.Series(y_test_encoded)
# Feature selection using XGBoost
xgb_selector = XGBClassifier(
  n_estimators=200,
  max_depth=6,
  learning_rate=0.1,
  subsample=0.8,
  colsample_bytree=0.8,
  use_label_encoder=False,
  eval_metric='logloss',
  random_state=43
)
sfm = SelectFromModel(estimator=xgb_selector, threshold='median')
X_train_selected = sfm.fit_transform(X_train, y_train)
X_test_selected = sfm.transform(X_test)
selected_features = [cat_cols[i] for i in sfm.get_support(indices=True)]
print(f"{len(selected_features)} features were selected:\n{selected_features}")
X_train = pd.DataFrame(X_train_selected, columns=selected_features)
X_test = pd.DataFrame(X_test_selected, columns=selected_features)
```

```
8 features were selected:
['exercise_frequency', 'mental_health_rating', 'netflix_hours', 'part_time_job', 'sleep_hours', 'social_media_hours',
'study_hours_per_day', 'media_usage_hours_pca']
```

The embedded feature selection with **XGBoost** identified eight key attributes balancing academic habits, lifestyle, and psychological factors. Applying a `threshold='median'` filter effectively reduced features to those with importance scores above the median, yielding a focused and robust subset for accurate modeling.

Interestingly, several of the eight features selected by the embedded **XGBoost** method were not among those showing strong visual or statistical correlations with academic performance during the data understanding phase. For example, attributes such as `exercise_frequency`, `social_media_hours`, and `sleep_hours` displayed weak or negligible associations with exam scores in the correlation matrix and scatter plots (see Section 3.1.2). Nonetheless, these features were retained by the model-driven selection process, which evaluates importance in the context of multivariate interactions and non-linear decision boundaries. This highlights the added value of using embedded methods, which can capture subtle, context-dependent patterns missed by univariate correlation analyses. As mentioned earlier, to maintain consistency and support XGBoost's expected performance, the selected subset was used to train both classifiers.

5.1.2 Classification models

A single decision tree was extracted from the trained Random Forest to examine its structure. This tree had a depth of 6 and contained 31 leaves, indicating a moderate level of complexity. Although individual trees in a Random Forest can overfit, the ensemble reduces this risk by averaging predictions across multiple trees. Visualizing such trees provides insight into decision-making processes, but validation metrics remain essential to assess generalization. The model was tuned with 300 estimators, a maximum depth of 6, and a 50% feature sampling rate per split, promoting moderately deep trees while limiting overfitting through restricted splits and leaf sizes.

Analysis of trees from the XGBoost model showed that they generally have shallower depths and fewer leaves than those in the Random Forest. One such tree had a depth of 5 with only 9 leaves, illustrating a simpler structure. This compactness supports better generalization and reduced overfitting. Unlike Random Forest, XGBoost grows trees sequentially to correct prior errors. Its best configuration included 400 estimators, a learning rate of 0.05, and regularization terms that further constrain complexity. These parameters encourage smaller, focused trees, highlighting how XGBoost balances efficiency and predictive accuracy through boosting.

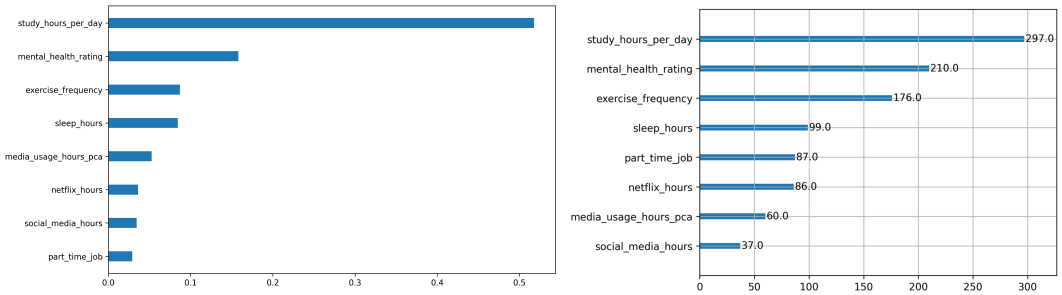

Figure 5.1: Feature importance plots: Random Forest (left) and XGBoost (right).

The feature importance analysis reveals both similarities and fundamental differences between Random Forest and XGBoost in how they evaluate feature relevance. In the Random Forest model, feature importance scores are *normalized* values that sum to 1, reflecting the *average decrease in impurity* (Gini importance) contributed by each feature across all trees. For instance, `study_hours_per_day` accounts for over half of the total importance (0.5178), indicating its dominant influence in the model. In contrast, XGBoost uses a *frequency-based metric* (when `importance_type='weight'`), counting the number of times a feature is used in any split across all trees. Here, `study_hours_per_day` was used 297 times, again emerging as the most frequently used feature, followed by `mental_health_rating` (210 times). While both models highlight the same top predictors, the scale and interpretation of their scores differ: Random Forest's are relative and normalized, whereas XGBoost's are raw counts reflecting frequency of usage in decision splits.

Now is the time for the optimized Random Forest and XGBoost classifiers to be compared in their ability to identify at-risk students. Modest gains in accuracy and improved `AtRisk` detection were anticipated due to XGBoost's boosting and regularization.

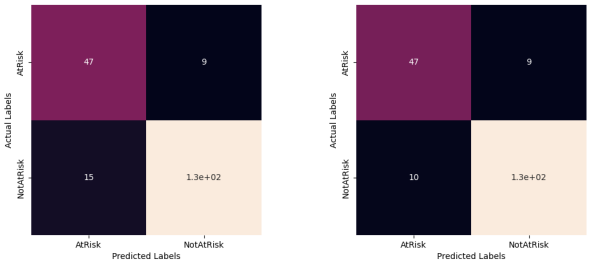

Figure 5.2: Confusion matrices on test set for Random Forest (left) and XGBoost (right).

Table 5.1: Classification performance metrics: Random Forest vs. XGBoost

Metric	**Random Forest Classifier**	**XGBoost Classifier**
Mean Accuracy	93.84% ± 0.87	**94.79% ± 0.47**
Mean Precision	94.03% ± 0.74	**94.86% ± 0.48**
Mean Recall	93.84% ± 0.87	**94.79% ± 0.47**
Mean F1-score	93.83% ± 0.88	**94.79% ± 0.47**
Class		
`AtRisk`	Precision: 0.76, Recall: **0.84**, F1-score: 0.80, Support: 56	Precision: **0.82**, Recall: **0.84**, F1-score: **0.83**, Support: 56
`NotAtRisk`	Precision: 0.93, Recall: 0.90, F1-score: 0.91, Support: 144	Precision: **0.94**, Recall: **0.93**, F1-score: **0.93**, Support: 144
Accuracy	0.88	**0.91**
Macro Avg	Precision: 0.85, Recall: 0.87, F1-score: 0.86	Precision: **0.88**, Recall: **0.88**, F1-score: **0.88**
Weighted Avg	Precision: 0.89, Recall: 0.88, F1-score: 0.88	Precision: **0.91**, Recall: **0.91**, F1-score: **0.91**

5.2 Review of classification models performance

Both models deliver strong classification performance, but XGBoost slightly outperforms Random Forest across most global metrics. On the test set, XGBoost achieves an accuracy of **94.79%**, precision of **94.86%**, recall of **94.79%**, and F1-score of **94.79%**, compared to Random Forest's 93.84%, 94.03%, 93.84%, and 93.83%, respectively. Class-level results show that while both models excel at identifying `NotAtRisk` students (precision 0.93–0.94, recall 0.90–0.93), XGBoost provides a notable improvement in identifying `AtRisk` students. It boosts precision from 0.76 to **0.82** and F1-score from 0.80 to **0.83**, indicating stronger focus on harder-to-classify minority instances. The confusion matrices in Figure 5.2 visually reinforce this performance gap between models.

Further comparison through aggregate metrics highlights XGBoost's advantage: macro-averaged precision, recall, and F1-score increase from 0.85–0.87 under Random Forest to **0.88**, while weighted averages improve from 0.88–0.89 to **0.91** (see Table 5.1). These improvements can be attributed to XGBoost's use of L1 (Lasso) and L2 (rigid) regularization and subsampling, which help reduce overfitting and improve generalization. Random Forest, using 300 trees with maximum depth 6 and 50% feature sampling, remains a strong contender but displays slightly more variance. For tasks prioritizing high predictive accuracy, particularly for the minority `AtRisk` class, XGBoost stands out as the superior model.

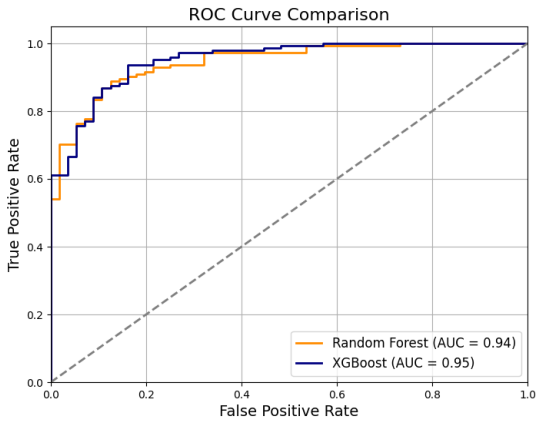

Figure 5.3: ROC curve comparison.

The AUC-ROC scores reinforce the overall superiority of XGBoost, with an area under the curve of **0.95** compared to **0.94** for Random Forest. Both values indicate excellent discriminatory power, but the slightly higher AUC of XGBoost suggests a more reliable distinction between `AtRisk` and `NotAtRisk` students across all classification thresholds.

Overview of hyperparameter tuning The XGBoost classifier is initialized with a fixed random state and the logloss evaluation metric to ensure consistent and meaningful model assessment. Its hyperparameter space includes the number of estimators from 100 to 700, balancing complexity and training time, with tree depths limited between 3 and 7 to avoid overfitting. Learning rates from 0.01 to 0.1 promote smooth convergence, while subsampling of rows and columns between 0.6 and 0.8 introduces randomness to reduce overfitting. The gamma parameter encourages pruning by requiring a minimum loss reduction for splits, and minimum child weight settings enforce conservative splitting. L1 and L2 regularization terms further control model complexity.

For the Random Forest classifier, the baseline of 100 trees is expanded up to 300 to improve stability through averaging. Maximum depths from 3 to 6 are explored to balance bias and variance. The `max_features` parameter varies between the square root of total features and fixed fractions to enhance tree diversity. Minimum sample split and leaf sizes prevent overfitting by ensuring sufficient data at splits and leaves. Both bootstrapped and full data sampling methods are tested to understand their effect on model bias and variance. This tuning strategy aims to find hyperparameters that yield accurate and generalizable models.

With the predictive models established and their performance thoroughly assessed, we now turn to summarizing the key findings, discussing practical implications, and reflecting on the study's limitations and potential avenues for future work in the concluding chapter.

Chapter 6

Evaluation

This study developed an early warning system to identify students at risk of academic difficulties weeks before final grades are available, by analyzing behavioral, lifestyle, and psychological data to support timely educator interventions. The model's effectiveness and key findings are discussed in Section 6.1, while important data considerations and limitations are examined in Section 6.2.

6.1 Model evaluation and key findings

Academic performance was found to be influenced by multiple factors beyond study time alone. Some students achieved high exam results despite limited study hours, often linked to strong mental health and fewer distractions. Conversely, success was observed among students facing mental health challenges, indicating that wellbeing and academic performance do not always align. These findings highlight the insufficiency of relying on only a few indicators to understand or predict risk effectively.

The predictive model performed well in identifying the majority of at-risk students while keeping false alarms to a minimum. It uses a practical combination of easy-to-measure factors, such as study habits, sleep patterns, mental health ratings, and media usage, making it both accurate and applicable in real educational settings.

A key insight from this study is the critical importance of thorough data understanding and preparation, as emphasized in the CRISP-DM framework. The analysis revealed that academic risk cannot Be accurately predicted by relying on limited or superficial data alone, deep exploration of behavioral, lifestyle, and psychological factors was essential. Careful handling of missing values, outliers, and data imbalances proved crucial in building a robust model. This experience underscores that successful data mining depends not just on modeling techniques but equally on the quality and readiness of the data.

6.2 Data considerations and limitations

However, some limitations remain. The dataset was relatively small and drawn from a single population, which may affect the model's accuracy when applied to larger or more diverse groups. Missing data were imputed using simple assumptions that might overlook important background differences. Additionally, including genuine but unusual cases improved detection of some high-risk students but slightly reduced overall prediction accuracy.

Overall, this evaluation demonstrates that data mining techniques can successfully integrate diverse student data to provide early, actionable insights. These insights can aid educators in targeting support effectively, but the model should be used alongside broader academic and wellbeing interventions. Future work should focus on expanding the dataset, incorporating more diverse factors, and validating the model across varied educational contexts to improve reliability and fairness.

Chapter 7

Conclusion and Perspectives

This study successfully developed an early warning system designed to identify students at risk of academic difficulties weeks before final grades are issued. By analyzing a rich combination of behavioral, lifestyle, and psychological data, the model goes beyond traditional predictors like study time to capture a more holistic view of student performance. The results demonstrated that academic success and wellbeing do not always align, emphasizing the importance of considering multiple, diverse factors when assessing risk. The predictive model performed strongly, accurately detecting most at-risk students while minimizing false positives. Importantly, the experience highlighted that effective data mining depends as much on thorough data exploration and preparation, following principles from the CRISP-DM framework. Careful management of missing data, outliers, and data imbalances was critical in building a reliable system.

Despite these strengths, several limitations were noted, including the relatively small and homogeneous dataset, which may limit generalizability across broader or more diverse populations. Simplified assumptions in data imputation and the inclusion of rare but valid outlier cases introduced trade-offs between sensitivity and overall accuracy. For practical deployment, this predictive tool should be integrated within broader academic and wellbeing support strategies to address the multifaceted nature of student success. Future research should aim to expand the dataset size and diversity, incorporate additional relevant factors such as tutoring and family support, and validate the model across varied educational settings. By continuing to refine and contextualize such systems, educators can be better equipped with timely, actionable insights to support student achievement and wellbeing effectively.

7. CONCLUSION AND PERSPECTIVES

References

Breiman, L. (2001). Random forests. Machine learning 45, 5–32. 27

Chen, T. and C. Guestrin (2016). XGBoost: A Scalable Tree Boosting System. In Proceedings of the 22nd acm sigkdd international conference on knowledge discovery and data mining, pp. 785–794. 27

Fayyad, U. M. and K. B. Irani (1993). Multi-interval discretization of continuous-valued attributes for classification learning. In Proceedings of the Thirteenth International Joint Conference on Artificial Intelligence (IJCAI), Volume 2, pp. 1022–1027. Morgan Kaufmann. 22

Han, H., W.-Y. Wang, and B.-H. Mao (2005). Borderline-SMOTE: a New Oversampling Method in Imbalanced Data Sets Learning. In International conference on intelligent computing, pp. 878–887. Springer. 23

Nath, J. (2025). Student habits vs academic performance. Dataset. URL: `https://www.kaggle.com/datasets/jayaantanaath/student-habits-vs-academic-performance`. 4, 5

Pearson, K. (1901). Liii. on lines and planes of closest fit to systems of points in space. The London, Edinburgh, and Dublin Philosophical Magazine and Journal of Science 2(11), 559–572. 21, 22

REFERENCES

Appendix A

Extract of codes

A.1 Skewness of numerical data

```
from scipy.stats import skew
print("\nSkewness (scipy):")
for col in sd.select_dtypes(include=['number']).columns:
print(f"{col}: {round(skew(sd[col], nan_policy='omit'), 4)}")
```

A.2 Correlations with ordinal attributes

```
sdc = sd.copy()
# Ordinal mapping for 'diet_quality'
diet_map = {'Poor': 1, 'Fair': 2, 'Good': 3}
sdc['diet_quality_ordinal'] = sdc['diet_quality'].map(diet_map)
# Handle 'parental_education_level' with 'Unknown' as separate category
sdc['parental_education_level'] = sdc['parental_education_level'].fillna('Unknown')
education_map = {'High School': 1, 'Bachelor': 2, 'Master': 3, 'Unknown': 0}
sdc['parental_education_ordinal'] = sdc['parental_education_level'].map(education_map)
# Ordinal mapping for 'internet_quality'
internet_map = {'Poor': 1, 'Average': 2, 'Good': 3}
sdc['internet_quality_ordinal'] = sdc['internet_quality'].map(internet_map)
# Spearman correlation between diet_quality_ordinal and exam_score
rho_diet_exam, pval_diet_exam = spearmanr(sdc['diet_quality_ordinal'], sdc['exam_score'])
print(f"Spearman's rho between diet_quality and exam_score = {rho_diet_exam:.3f}, p-value = {pval_diet_exam:.4f}")
# Spearman correlation between parental_education_ordinal and exam_score
rho_education_exam, pval_education_exam = spearmanr(sdc['parental_education_ordinal'], sdc['exam_score'])
print(f"Spearman's rho between parental_education and exam_score = {rho_education_exam:.3f}, p-value = {pval_education_exam:.4f}")
# Spearman correlation between internet_quality_ordinal and exam_score
rho_internet_exam, pval_internet_exam = spearmanr(sdc['internet_quality_ordinal'], sdc['exam_score'])
print(f"Spearman's rho between internet_quality and exam_score = {rho_internet_exam:.3f}, p-value = {pval_internet_exam:.4f}")
# Kendall's tau correlation between diet_quality_ordinal and exam_score
tau_diet_exam, pval_tau_diet_exam = kendalltau(sdc['diet_quality_ordinal'], sdc['exam_score'])
print(f"Kendall's tau between diet_quality and exam_score = {tau_diet_exam:.3f}, p-value = {pval_tau_diet_exam:.4f}")
# Kendall's tau correlation between parental_education_ordinal and exam_score
tau_education_exam, pval_tau_education_exam = kendalltau(sdc['parental_education_ordinal'], sdc['exam_score'])
print(f"Kendall's tau between parental_education and exam_score = {tau_education_exam:.3f}, p-value = {pval_tau_education_exam:.4f}")
# Kendall's tau correlation between internet_quality_ordinal and exam_score
tau_internet_exam, pval_tau_internet_exam = kendalltau(sdc['internet_quality_ordinal'], sdc['exam_score'])
print(f"Kendall's tau between internet_quality and exam_score = {tau_internet_exam:.3f}, p-value = {pval_tau_internet_exam:.4f}")
```

```
Spearman's rho between diet_quality and exam_score = 0.014, p-value = 0.6634
Spearman's rho between parental_education and exam_score = -0.022, p-value = 0.4950
Spearman's rho between internet_quality and exam_score = -0.046, p-value = 0.1479
Kendall's tau between diet_quality and exam_score = 0.010, p-value = 0.6739
Kendall's tau between parental_education and exam_score = -0.016, p-value = 0.4944
Kendall's tau between internet_quality and exam_score = -0.035, p-value = 0.1538
```

```python
# Group-by summary of 'exam_score' based on each of these ordinal variables
summary_diet = sdc.groupby('diet_quality_ordinal')['exam_score'] \
.agg(count='count', median='median', iqr=lambda x: x.quantile(0.75)-x.quantile(0.25))
print("\nExam score by diet quality (median & IQR):")
print(summary_diet)
summary_education = sdc.groupby('parental_education_ordinal')['exam_score'] \
.agg(count='count', median='median', iqr=lambda x: x.quantile(0.75)-x.quantile(0.25))
print("\nExam score by parental education level (median & IQR):")
print(summary_education)
summary_internet = sdc.groupby('internet_quality_ordinal')['exam_score'] \
.agg(count='count', median='median', iqr=lambda x: x.quantile(0.75)-x.quantile(0.25))
print("\nExam score by internet quality (median & IQR):")
print(summary_internet)
```

```
Exam score by diet quality (median & IQR):
                     count  median    iqr
diet_quality_ordinal
1                      185    67.4  22.00
2                      437    71.0  23.00
3                      378    70.3  22.85


Exam score by parental education level (median & IQR):
                           count  median     iqr
parental_education_ordinal
0                             91   71.00  21.400
1                            392   69.95  24.025
2                            350   71.70  23.200
3                            167   66.80  21.900


Exam score by internet quality (median & IQR):
                         count  median     iqr
internet_quality_ordinal
1                          162    71.4  21.675
2                          391    71.0  25.950
3                          447    69.3  21.350
```

A.3 Statistical analyses

```python
# Code 1
# Define the two groups for the t-test
male_scores = sd[sd['gender'] == 'Male']['exam_score']
female_scores = sd[sd['gender'] == 'Female']['exam_score']
# Perform a two-sample t-test
t_stat, p_value = stats.ttest_ind(male_scores, female_scores, nan_policy='omit')
# Print the hypotheses explicitly
print("Null hypothesis (H0): There is no significant difference in exam scores between males and females.")
print("Alternative hypothesis (H1): There is a significant difference in exam scores between males and females.")
# Print the results of the t-test
print("\nT-statistic:", t_stat)
print("P-value:", p_value)
# Interpretation
alpha = 0.05 # Significance level (5%)
if p_value < alpha:
  print("\nReject the null hypothesis: There is a significant difference in exam scores between males and females.")
else:
  print("\nFail to reject the null hypothesis: There is no significant difference in exam scores between males and females.")
```

```
Null hypothesis (H0): There is no significant difference in exam scores between males and females.
Alternative hypothesis (H1): There is a significant difference in exam scores between males and females.
T-statistic: -0.33908014636930084
P-value: 0.734623820821068
Fail to reject the null hypothesis: There is no significant difference in exam scores between males and females.
```

```python
# Code 2: Combining attributes, applying binning, and performing One-Way ANOVA
# Step 1: Combine social media hours and Netflix hours into a new variable 'total_media_usage'
sd['total_media_usage'] = sd['social_media_hours'] + sd['netflix_hours']
# Step 2: Create the 'media_usage' categorical variable based on distribution
# Using the summary stats from the provided data for setting thresholds
low_threshold = sd['total_media_usage'].quantile(0.25) # 25th percentile
high_threshold = sd['total_media_usage'].quantile(0.75) # 75th percentile
# Define the 'media_usage' categories
sd['media_usage'] = pd.cut(sd['total_media_usage'],
                        bins=[-float('inf'), low_threshold, high_threshold, float('inf')],
                        labels=['Low', 'Moderate', 'High'])
# Step 3: Perform One-Way ANOVA to test if exam scores differ across the 'media_usage' categories
# Group exam scores by 'media_usage'
low_media_scores = sd[sd['media_usage'] == 'Low']['exam_score']
moderate_media_scores = sd[sd['media_usage'] == 'Moderate']['exam_score']
high_media_scores = sd[sd['media_usage'] == 'High']['exam_score']
# Perform one-way ANOVA
f_stat, p_value = stats.f_oneway(low_media_scores, moderate_media_scores, high_media_scores)
# Print the hypotheses explicitly
print("Null Hypothesis (H0): There is no significant difference in exam scores between the media usage categories.")
print("Alternative Hypothesis (H1): There is a significant difference in exam scores between the media usage categories.")
# Print the results of the ANOVA
print("\nF-statistic:", f_stat)
print("P-value:", p_value)
# Interpretation
alpha = 0.05 # Significance level (5%)
if p_value < alpha:
   print("\nReject the null hypothesis: There is a significant difference in exam scores between the media usage categories.")
else:
   print("\nFail to reject the null hypothesis: There is no significant difference in exam scores between the media usage categories.")
```

```
Null Hypothesis (H0): There is no significant difference in exam scores between the media usage categories.
Alternative Hypothesis (H1): There is a significant difference in exam scores between the media usage categories.
F-statistic: 26.17473515160436
P-value: 8.333795224890345e-12
Reject the null hypothesis: There is a significant difference in exam scores between the media usage categories.
```

A.4 Dealing with missing data

```python
from sklearn.impute import KNNImputer
sd_XX = sd_X.copy()
sd_XX.drop(columns='student_id', inplace=True)
sd_XX = sd_XX.drop(columns=['exam_score', 'grade'])
cat_cols = list(sd_XX.select_dtypes(include=['object']).columns)
ordinal_encoder = OrdinalEncoder()
sd_XX[cat_cols] = ordinal_encoder.fit_transform(sd_XX[cat_cols])
# Fill missing value in using KNN imputer from scikit-learn
imputer = KNNImputer() # default n_neighbors=5, n_neighbors
# n_neighbors 3 and 7 where tested an the performace declined compared to the previous mode imputation
sd_XX[list(sd_XX.columns)] = imputer.fit_transform(sd_XX[list(sd_XX.columns)])
clf = DecisionTreeClassifier(random_state=43)
ad.cross_validation_avg_scores(sd_XX, sd_y, clf)
```

```
Mean Accuracy is 84.70 +/- 2.64
Mean Precision is 81.16 +/- 3.30
Mean Recall is 80.86 +/- 2.85
Mean F-score is 80.99 +/- 3.06
```

```
sd_XX = sd_X.copy()
sd_XX.drop(columns='student_id', inplace=True)
sd_XX = sd_XX.drop(columns=['exam_score', 'grade'])
# Impute missing values in 'parental_education_level' with the mode
sd_XX['parental_education_level'] = sd_XX['parental_education_level'].fillna(sd_XX['parental_education_level'].mode()[0])
cat_cols = list(sd_XX.select_dtypes(include=['object']).columns)
ordinal_encoder = OrdinalEncoder()
sd_XX[cat_cols] = ordinal_encoder.fit_transform(sd_XX[cat_cols])
clf = DecisionTreeClassifier(random_state=43, class_weight='balanced')
ad.cross_validation_avg_scores(sd_XX, sd_y, clf)
```

```
Mean Accuracy is 85.80 +/- 1.72
Mean Precision is 82.91 +/- 2.49
Mean Recall is 81.41 +/- 1.69
Mean F-score is 82.04 +/- 1.94
```

A.5 Outliers

```
from sklearn.preprocessing import StandardScaler
from sklearn.decomposition import PCA
from sklearn.ensemble import IsolationForest
# Select numeric features for outlier detection
features = ['study_hours_per_day', 'exam_score', 'sleep_hours', 'social_media_hours',
            'mental_health_rating', 'attendance_percentage', 'exercise_frequency']
# Use 'sd' as the dataset name
df = sd[features].dropna()
scaler = StandardScaler()
df_scaled = scaler.fit_transform(df)
# Apply Isolation Forest
iso = IsolationForest(contamination=0.025, random_state=42)
outlier_pred = iso.fit_predict(df_scaled) # -1 = outlier, 1 = inlier
# Reduce dimensions for visualization using PCA
pca = PCA(n_components=2)
df_pca = pca.fit_transform(df_scaled)
# Create a DataFrame for plotting and row extraction
outlier_df = sd.loc[df.index].copy()
outlier_df['PCA1'] = df_pca[:, 0]
outlier_df['PCA2'] = df_pca[:, 1]
outlier_df['outlier_flag'] = outlier_pred
# Show outliers only (text output)
outliers_only = outlier_df[outlier_df['outlier_flag'] == -1]
print(f"Number of outliers detected: {len(outliers_only)}")
print(outliers_only[['exam_score', 'study_hours_per_day', 'mental_health_rating', 'social_media_hours']].to_string())

# Plot: outliers in red
plt.figure(figsize=(8, 6))
plt.scatter(outlier_sd['PCA1'], outlier_sd['PCA2'],
          c=outlier_sd['Outlier'].map({1: 'blue', -1: 'red'}),
          alpha=0.6, label='Outliers')
plt.title('Multivariate outlier detection with Isolation Forest')
plt.xlabel('PCA1')
plt.ylabel('PCA2')
plt.grid(True)
plt.legend()

# Save the figure to a file
plt.savefig('img/outlier_detection_plot.png', format='png')

# Show the plot
plt.show()
```

```
Number of outliers detected: 25
     exam_score  study_hours_per_day  mental_health_rating  social_media_hours
...
92         43.9                  0.5                     3                 0.8
...
447        93.6                  5.9                     1                 0.0
...
712        98.7                  3.0                    10                 0.0
...
998        69.7                  5.4                     1                 4.1
```

A.6 Preprocessing validation pipeline

```
# Apply PCA on the original numerical data (social_media_hours, netflix_hours)
sd_XX = sd_X.copy()
pca = PCA(n_components=1)
sd_XX['media_usage_hours_pca'] =
pca.fit_transform(sd_XX[['social_media_hours', 'netflix_hours']])

# Remove the proxy columns to avoid data leakage
sd_XX.drop(columns=['student_id', 'exam_score', 'grade'], inplace=True)

# Train-test split
X_train, X_test, y_train, y_test = train_test_split(
   sd_XX, sd_y, test_size=0.2, random_state=43, stratify=sd_y
)

# Better discretization settings: more robust grid
dt_binner = DecisionTreeDiscretiser(
   bin_output='boundaries', cv=5, precision=2,
   scoring='accuracy', regression=False, random_state=43,
   param_grid={'max_depth': [3, 4, 5, 6], 'ccp_alpha': [0.0, 0.005, 0.01]}
)

# Apply discretization
X_train_binned = dt_binner.fit_transform(X_train, y_train)
X_test_binned = dt_binner.transform(X_test)
X_train = pd.DataFrame(X_train_binned)
X_test = pd.DataFrame(X_test_binned)
y_train = pd.Series(y_train)

# Categorical encoding
cat_cols = X_train.select_dtypes(include=['object']).columns.tolist()
preprocessor = Pipeline([
   ('imputer', SimpleImputer(strategy='most_frequent')),
   ('encoder', OrdinalEncoder())
])
X_train = pd.DataFrame(preprocessor.fit_transform(X_train[cat_cols]), columns=cat_cols)
X_test = pd.DataFrame(preprocessor.transform(X_test[cat_cols]), columns=cat_cols)

# BorderlineSMOTE to handle imbalance
smote = BorderlineSMOTE(random_state=43, k_neighbors=5, m_neighbors=10)
#smote = SMOTE(random_state=43)
X_train, y_train = smote.fit_resample(X_train, y_train)

# Scaling
scaler = StandardScaler()
X_train = pd.DataFrame(scaler.fit_transform(X_train), columns=cat_cols)
X_test = pd.DataFrame(scaler.transform(X_test), columns=cat_cols)
```

```
# Feature selection
selector_estimator = DecisionTreeClassifier(max_depth=7, min_samples_split=4,min_samples_leaf=10, ccp_alpha=0.007,
                                            random_state=43)
sfm = SelectFromModel(estimator=selector_estimator, threshold='median')
X_train_selected = sfm.fit_transform(X_train, y_train)
X_test_selected = sfm.transform(X_test)
selected_features = [cat_cols[index] for index in sfm.get_support(indices=True)]
X_train = pd.DataFrame(X_train_selected, columns=selected_features)
X_test = pd.DataFrame(X_test_selected, columns=selected_features)
# Cross-validation setup
stratified_cv = StratifiedKFold(n_splits=5, shuffle=True, random_state=43)
# Extended search space for better tuning
params_dist = {
  'splitter': ['best'],
  'criterion': ['gini', 'entropy'],
  'max_depth': range(5, 7),
  'min_impurity_decrease': np.linspace(0.005, 0.05, 10),
  'min_samples_leaf': range(4, 16, 2),
  'min_samples_split': range(4, 12, 2),
  'ccp_alpha': uniform(loc=0.005, scale=0.004)
}
clf = DecisionTreeClassifier(random_state=43)
random_search = RandomizedSearchCV(
   clf,
   param_distributions=params_dist,
   n_iter=300,
   cv=stratified_cv,
   random_state=43,
   n_jobs=-1,
   scoring='accuracy'
)
random_search.fit(X_train, y_train)
# Evaluate best classifier
print(f'Best parameters found: {random_search.best_params_}')
best_clf = random_search.best_estimator_
# Evaluation
ad.cross_validation_avg_scores(X_train, y_train, best_clf, cv_=5, print_dict=False)
y_pred = best_clf.predict(X_test)
print(ad.classification_report(y_test, y_pred))
ad.plot_save_confusion_matrix(y_test, y_pred, "confusion_matrix_dt")
```

A.7 Random Forest classifier

```
# Random Forest classifier
clf = RandomForestClassifier(n_estimators=100, random_state=43)
param_dist = {
  'n_estimators': [100, 200, 300], # More trees reduce variance but increase training time.
  'max_depth': [3, 4, 5, 6], # Limits tree depth to prevent overfitting or underfitting.
  'max_features': ['sqrt', 0.5, 0.7], # Controls randomness by limiting features considered at splits.
  'min_samples_split': [3, 5, 7, 9], # Prevents splits on small samples to reduce overfitting.
  'min_samples_leaf': [2, 4, 6], # Ensures leaf nodes have enough samples for reliable decisions.
  'bootstrap': [True, False] # Enables randomness with bootstrapping or uses full data per tree.
}

stratified_cv = StratifiedKFold(n_splits=5, shuffle=True, random_state=43)
random_search = RandomizedSearchCV(
   clf,
   param_distributions=param_dist,
   n_iter=300,
   cv=stratified_cv,
   verbose=1,
   random_state=43,
   n_jobs=-1
)
random_search.fit(X_train, y_train)
```

```
# Evaluation
print(f'Best parameters found: {random_search.best_params_}')
best_clf_rf = random_search.best_estimator_
ad.cross_validation_avg_scores(X_train, y_train, best_clf_rf, cv_=5, print_dict=False)
y_pred_encoded = best_clf_rf.predict(X_test)
y_pred = label_encoder.inverse_transform(y_pred_encoded)
y_test = label_encoder.inverse_transform(y_test)
print(ad.classification_report(y_test, y_pred))
ad.plot_save_confusion_matrix(y_test, y_pred, "confusion_matrix_rf")
```

A.8 XGBoost classifier

```
# XGBoost classifier
clf = XGBClassifier(
  random_state=43,
  eval_metric='logloss',
)
# Hyperparameter space for XGBoost (for RandomizedSearchCV)
param_dist = {
  'n_estimators': np.arange(100, 701, 100), # Reduce upper limit to avoid too many trees
  'max_depth': [3, 4, 5, 6, 7], # Shallow trees generalize better
  'learning_rate': [0.01, 0.05, 0.1], # Smaller values for smoother learning
  'subsample': [0.6, 0.7, 0.8], # Avoid overfitting by using row sampling
  'colsample_bytree': [0.6, 0.7, 0.8], # Avoid overfitting by using feature sampling
  'gamma': [0.1, 0.3, 0.5, 1], # Minimum gain needed to split a node (adds pruning)
  'min_child_weight': [3, 5, 7], # Minimum sum of instance weight in a child (larger = more conservative)
  'reg_alpha': [0.01, 0.1, 1], # L1 regularization
  'reg_lambda': [1, 2, 5] # L2 regularization
}
# Define the Stratified K-Fold cross-validator
stratified_cv = StratifiedKFold(n_splits=5, shuffle=True, random_state=43)
# RandomizedSearchCV for hyperparameter tuning
random_search = RandomizedSearchCV(
  clf, param_distributions=param_dist,
  n_iter=500,
  cv=stratified_cv,
  verbose=1,
  random_state=43,
  n_jobs=-1
)
# Evaluation
random_search.fit(X_train, y_train)
# Print best hyperparameters
print(f'Best parameters found: {random_search.best_params_}')
best_clf_xgb = random_search.best_estimator_
ad.cross_validation_avg_scores(X_train, y_train, best_clf_xgb, cv_=5, print_dict=False)
y_pred = best_clf_xgb.predict(X_test)
# Decode the predictions and true values back to the original labels
y_pred = label_encoder.inverse_transform(y_pred)
y_test = label_encoder.inverse_transform(y_test)
# Report and Confusion Matrix
print(ad.classification_report(y_test, y_pred))
ad.plot_save_confusion_matrix(y_test, y_pred, "confusion_matrix_xgb")
```