Student Habits and Performance Analysis Application

1	23F-0636	Leader Muhammad Rameez	BCS-4B
2	23F-0674	M Kamran Ali	BCS-4B

Submited to : Dr. Haris Khurram

1. Problem Statement

In the realm of academic achievement, student performance is often influenced by a wide array of behavioral, lifestyle, and psychological factors. While educational institutions primarily focus on curriculum and pedagogy, there remains a lack of data-driven tools that can analyze how personal habits ,such as study duration, sleep quality, diet, and social media usage impact academic outcomes. This gap hinders educators, students, and policymakers from making informed decisions to enhance student success.

The problem this project addresses is the absence of an integrated, interactive system that can analyze, visualize, and predict academic performance based on student habits using modern statistical methods and machine learning models. By bridging this gap, the application aims to uncover meaningful insights into which habits most strongly correlate with academic success, thereby enabling students to optimize their behaviors and institutions to develop targeted interventions.

2. Objective

The primary objective of this project is to design and develop an interactive data analysis application that explores the relationship between student habits and academic performance using statistical and machine learning techniques. This application aims to provide insightful, evidence-based interpretations to help students, educators, and researchers better understand the key factors influencing academic success.
The specific objectives include:

- To collect and preprocess student behavioral data for statistical analysis.
- To analyze categorical and numerical student habit variables using visual tools such as pie charts, box plots, and distribution plots.
- To implement regression models for predicting academic outcomes like exam scores and attendance based on lifestyle habits (e.g., study hours, sleep duration).
- To explore the probability distributions of student performance metrics and behaviors using normal, binomial, Poisson, and uniform models.
- To compute and visualize confidence intervals for statistical predictions to enhance the reliability of the analysis.
- To present data in a user-friendly, interactive PyQt5 interface that enables non-technical users to explore and interpret results in real time.

3. Data Description

3.1.Data Source

The dataset used for this project was sourced from Kaggle, a popular online platform for data science and machine learning resources. The dataset focuses on various student habits and their potential impact on academic performance, providing a solid foundation for statistical and predictive analysis.

- **Source Platform:** **Kaggle**
- **Dataset Title:** Student Habits vs Academic Performance

3.2.Dataset Name

Student Study, Sleep, Social Media & Exam Performance

Link to Dataset

https://www.kaggle.com/datasets/jayaantanaath/student-habits-vs-academic-performance

3.3.Dataset Description

The dataset consists of **1,000 records** and **16 columns**, representing individual students along with various attributes of their behavior and academic scores. It provides a diverse range of lifestyle factors including study hours, media consumption, sleep, diet, physical activity, and more.

Column Name	Description
student_id	Unique identifier for each student
age	Age of the student
gender	Gender of the student (Male/Female)
study_hours_per_day	Average number of hours spent studying each day
social_media_hours	Average daily time spent on social media (in hours)
netflix_hours	Average daily time spent watching Netflix (in hours)
part_time_job	Whether the student has a part-time job (Yes/No)
attendance_percentage	Class attendance percentage
sleep_hours	Average number of hours slept daily
diet_quality	Self-assessed diet quality (e.g., Poor, Fair, Good)
exercise_frequency	Number of exercise sessions per week
parental_education_level	Highest education level of the student’s parents (e.g., High School, Master)
internet_quality	Self-rated quality of internet access (Poor/Average/Good)

Column Name	Description
mental_health_rating	Self-rated mental health on a scale of 1 to 10
extracurricular_participation	Whether the student participates in extracurricular activities (Yes/No)
exam_score	Final academic exam score (0–100 scale)

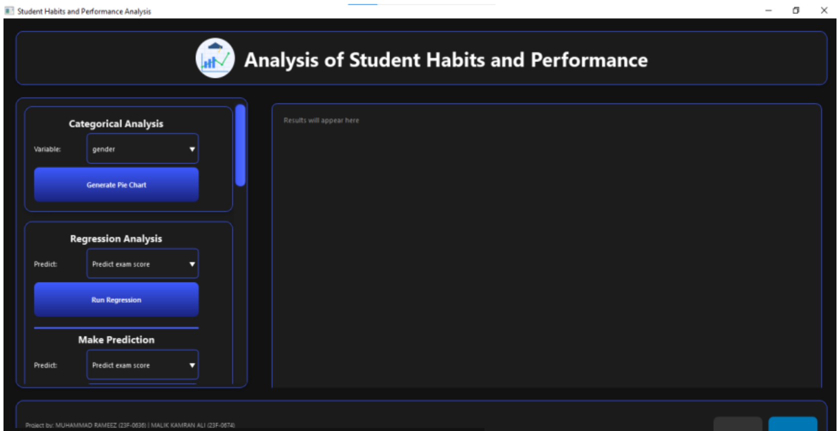
3.4.Data Format

- **File Format:** CSV (Comma-Separated Values)
- **File Name:** student_habits_performance.csv
- **Number of Records:** 1,000
- **Number of Columns:** 16
- **Data Types:**
  - Numerical: age, study_hours_per_day, sleep_hours, exam_score, etc.
  - Categorical: gender, diet_quality, part_time_job, etc.

3.5.Data Preprocessing

To ensure accurate and meaningful analysis, the following preprocessing steps were performed:

1. **Missing Value Handling**: The dataset was checked for null or missing entries; any such values were appropriately handled or removed.
2. **Categorical Encoding**: Variables like gender, diet_quality, and internet_quality were encoded numerically for model compatibility.
3. **Outlier Detection**: Visual tools like box plots and statistical thresholds were used to detect and treat extreme outliers.
4. **Normalization**: Features such as study_hours_per_day, social_media_hours, and exam_score were normalized or scaled for regression analysis.
5. **Data Type Validation**: Ensured all columns had appropriate and consistent data types (e.g., float, int, string).

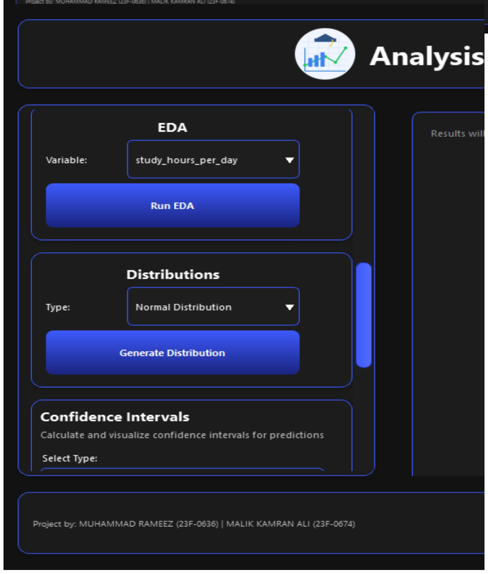


Variable: gender

Predict: Predict exam score

Predict: Predict exam score

Project by: MUHAMMAD RAMEEZ (23F-0636) | MALIK KAMRAN ALI (23F-0674)



Variable: study_hours_per_day

Type: Normal Distribution

Project by: MUHAMMAD RAMEEZ (23F-0636) | MALIK KAMRAN ALI (23F-0674)

4. Results

Main Interface Overview

Description:
This screenshot displays the **main dashboard** of the *Student Habits and Performance Analysis* application. The user interface is designed using **PyQt5** and features a clean, modern layout with a dark theme for enhanced readability.

On the left sidebar, there are three main functional areas:

1. **Categorical Analysis:**
Users can select categorical variables (e.g., gender, diet quality) and generate pie charts to visually understand distribution patterns.

2. **Regression Analysis:**
This section allows users to select a prediction type (such as "Predict exam score") and run a linear regression analysis to observe relationships between variables.

3. **Make Prediction:**
This module offers real-time predictions based on user input (e.g., entering study hours to predict exam scores).

4. **Exploratory Data Analysis (EDA):**
Users can select a numerical variable (in this case, study_hours_per_day) to analyze.
Clicking on “Run EDA” computes and visualizes key statistical metrics such as mean, median, variance, quartiles, and outliers.
Helpful for understanding data spread and identifying trends.

5. **Distributions Module:**
Offers several statistical distributions like **Normal**, **Binomial**, **Poisson**, and **Uniform**.
In this example, "Normal Distribution" is selected.

Clicking “Generate Distribution” displays a plot comparing the selected variable’s distribution to the expected theoretical distribution.

6. **Confidence Intervals Section:**

Used to compute **prediction intervals** and **95% confidence intervals** for regression models.

Helps in validating how confidently predictions (e.g., exam scores based on study hours) fall within a statistical range.

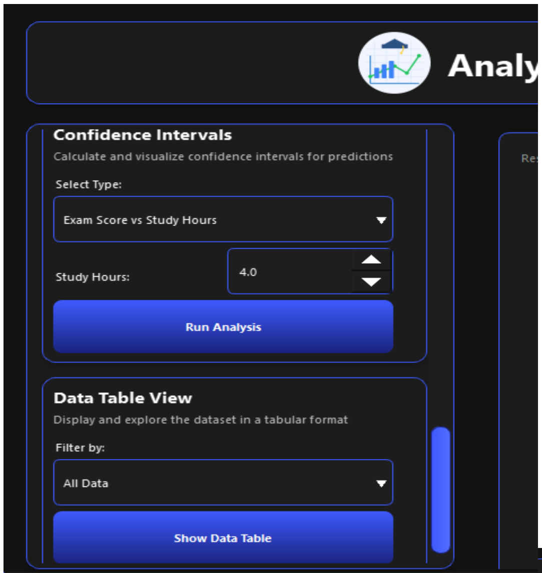
7. **Data Table View:**

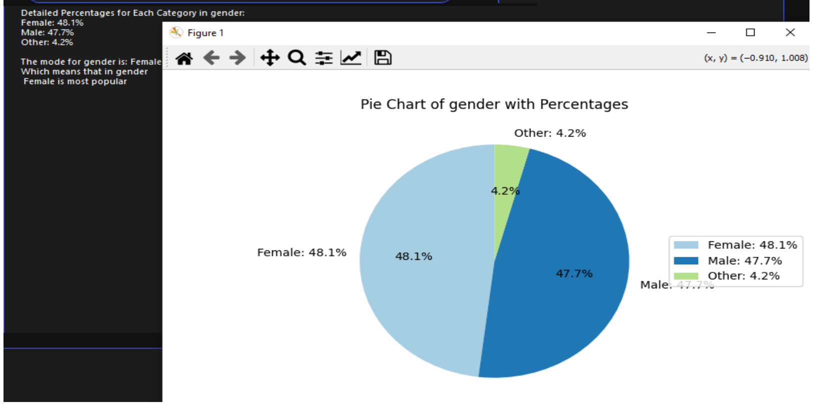
Displays the dataset in a tabular format.

Offers filtering options (e.g., "All Data").

On the right side is the Results Panel, where all outputs including charts, statistical values, and predictions will be displayed.

At the bottom, the developers’ names and GitHub/LinkedIn links are provided, enhancing the professional presentation of the application.


Analy
Confidence Intervals
Calculate and visualize confidence intervals for predictions
Select Type:
Exam Score vs Study Hours
Study Hours:
4.0
Run Analysis
Data Table View
Display and explore the dataset in a tabular format
Filter by:
All Data
Show Data Table
Re



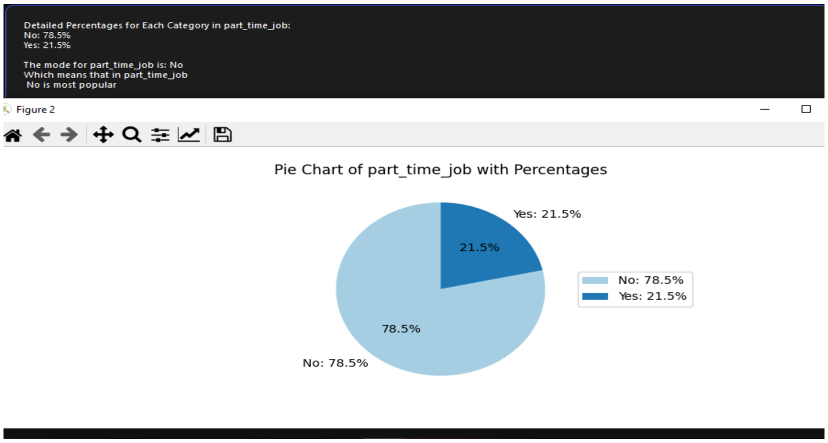
```
Detailed Percentages for Each Category in gender:
Female: 48.1%
Male: 47.7%
Other: 4.2%

The mode for gender is: Female
Which means that in gender
 Female is most popular
```

Pie Chart of Gender Distribution with Percentages

This pie chart visually represents the distribution of gender within the dataset. It consists of three segments, each representing a different gender category and its corresponding percentage:

- **Female:** Represented by a light blue segment, making up **48.1%** of the dataset.
- **Male:** Represented by a dark blue segment, constituting **47.7%** of the dataset.
- **Other:** Represented by a light green segment, accounting for **4.2%** of the dataset.

The percentages are also displayed directly on each segment of the pie chart, providing a clear view of the proportion of each gender within the analyzed data. The legend on the right-hand side further clarifies which color corresponds to each gender category and its respective percentage.



```
Detailed Percentages for Each Category in part_time_job:
No: 78.5%
Yes: 21.5%

The mode for part_time_job is: No
Which means that in part_time_job
 No is most popular
```

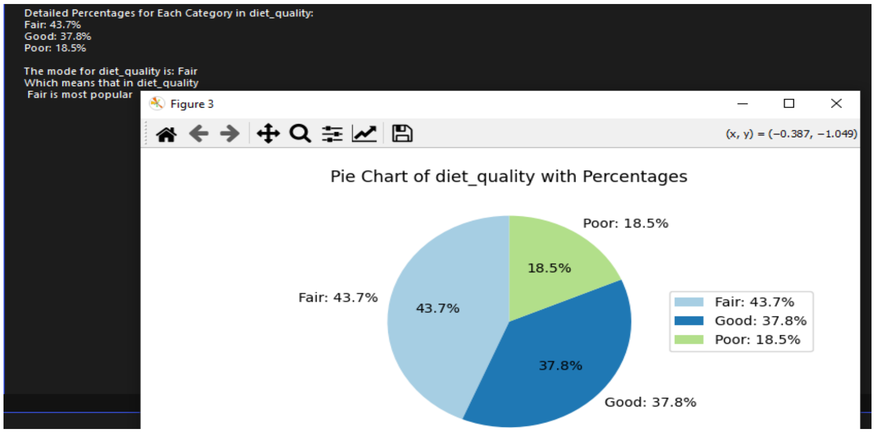
Pie Chart of Part-Time Job with Percentage

This pie chart illustrates the distribution of students who have a part-time job versus those who do not. The chart is divided into two segments:

- **No:** Represented by a light blue segment, indicating that **78.5%** of the students in the dataset do not have a part-time job.

- **Yes:** Represented by a dark blue segment, showing that **21.5%** of the students in the dataset have a part-time job.

The corresponding percentages are clearly labeled on each segment of the pie chart. A legend on the right provides a key to the colors and their respective categories and percentages.



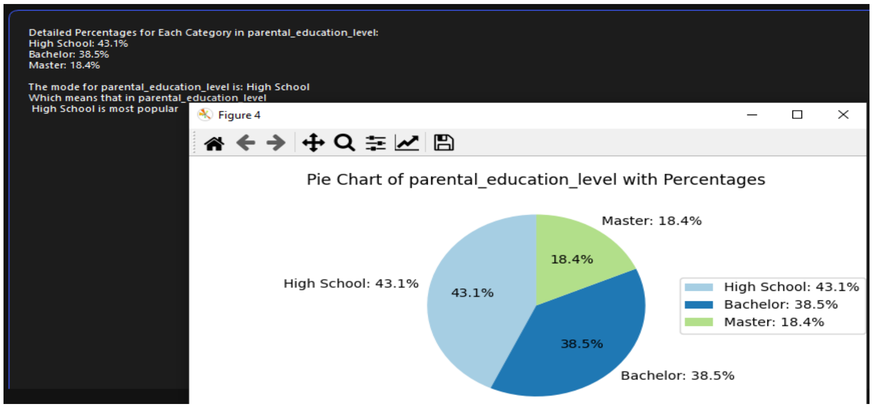
```
Detailed Percentages for Each Category in diet_quality:
Fair: 43.7%
Good: 37.8%
Poor: 18.5%

The mode for diet_quality is: Fair
Which means that in diet_quality
 Fair is most popular
```

Pie Chart of Diet Quality with Percentages

This pie chart displays the distribution of diet quality among the students in the dataset. The chart is divided into three segments, each representing a different level of diet quality and its corresponding percentage:

- **Fair:** Represented by a light blue segment, indicating that **43.7%** of the students have a fair quality diet.
- **Good:** Represented by a dark blue segment, showing that **37.8%** of the students have a good quality diet.
- **Poor:** Represented by a light green segment, accounting for **18.5%** of the students having a poor quality diet.

The percentage for each diet quality level is clearly marked on its respective segment in the pie chart. Additionally, a legend on the right provides a key to the colors and their corresponding diet quality categories and percentages.



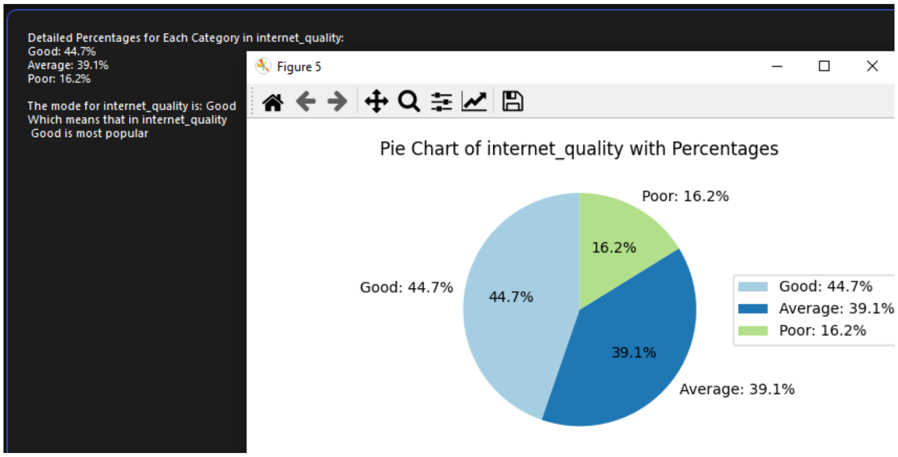
```
Detailed Percentages for Each Category in parental_education_level:
High School: 43.1%
Bachelor: 38.5%
Master: 18.4%

The mode for parental_education_level is: High School
Which means that in parental_education_level
 High School is most popular
```

Pie Chart of Parental Education Level with Percentages

This pie chart illustrates the distribution of the highest education level attained by the parents in the dataset. The chart is divided into three segments, each representing a different education level and its corresponding percentage:

- **High School:** Represented by a light blue segment, indicating that for **43.1%** of the students, the highest education level of their parents is high school.
- **Bachelor:** Represented by a dark blue segment, showing that for **38.5%** of the students, the highest education level of their parents is a Bachelor's degree.
- **Master:** Represented by a light green segment, accounting for **18.4%** of the students whose parents' highest education level is a Master's degree.

The percentage for each parental education level is clearly displayed on its respective segment within the pie chart. A legend on the right provides a key to the colors and their corresponding education levels and percentages.



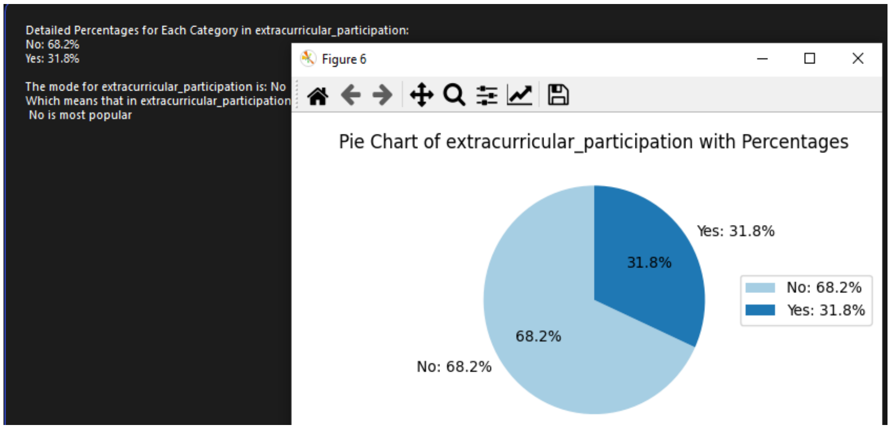
```
Detailed Percentages for Each Category in internet_quality:
Good: 44.7%
Average: 39.1%
Poor: 16.2%

The mode for internet_quality is: Good
Which means that in internet_quality
 Good is most popular
```

Pie Chart of Internet Quality with Percentages

This pie chart illustrates the distribution of internet quality experienced by the students in the dataset. The chart is divided into three segments, each representing a different level of internet quality and its corresponding percentage:

- **Good:** Represented by a light blue segment, indicating that **44.7%** of the students report having good quality internet.
- **Average:** Represented by a dark blue segment, showing that **39.1%** of the students experience average quality internet.
- **Poor:** Represented by a light green segment, accounting for **16.2%** of the students who report having poor quality internet.

The percentage for each level of internet quality is clearly marked on its respective segment within the pie chart. A legend on the right provides a key to the colors and their corresponding internet quality levels and percentages.



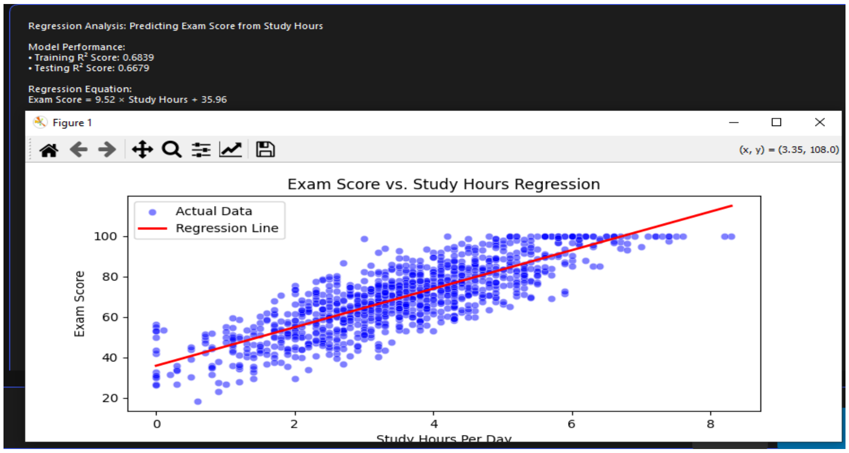
```
Detailed Percentages for Each Category in extracurricular_participation:
No: 68.2%
Yes: 31.8%

The mode for extracurricular_participation is: No
Which means that in extracurricular_participation
 No is most popular
```

Pie Chart of Extracurricular Participation with Percentages

This pie chart illustrates the distribution of students based on their participation in extracurricular activities. The chart is divided into two segments:

- **No:** Represented by a light blue segment, indicating that **68.2%** of the students in the dataset do not participate in extracurricular activities.
- **Yes:** Represented by a dark blue segment, showing that **31.8%** of the students in the dataset do participate in extracurricular activities.

The corresponding percentages are clearly labeled on each segment of the pie chart. A legend on the right provides a key to the colors and their respective categories and percentages.



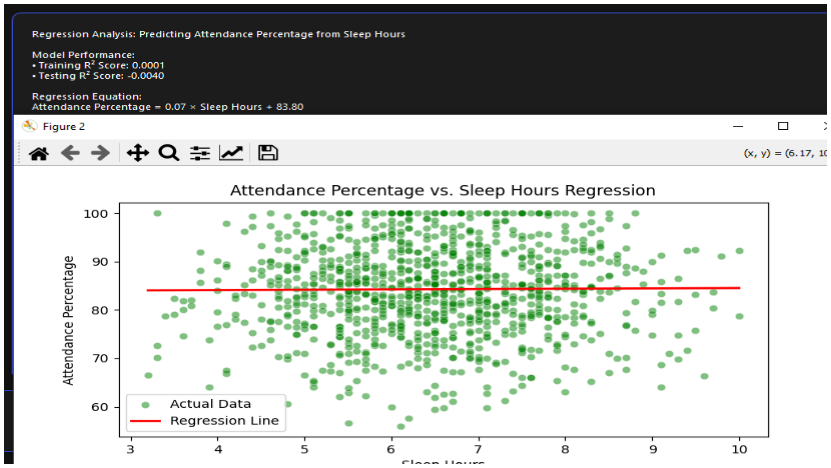
```
Regression Analysis: Predicting Exam Score from Study Hours

Model Performance:
• Training R² Score: 0.6839
• Testing R² Score: 0.6679

Regression Equation:
Exam Score = 9.52 × Study Hours + 35.96
```

Exam Score vs. Study Hours Regression

This scatter plot visualizes the relationship between the number of study hours per day and the corresponding exam scores of students. Each blue dot on the plot represents a single student and their respective data points for study hours and exam score.

A red line, labeled as the "Regression Line," is superimposed on the scatter plot. This line represents the best linear fit to the data, indicating the general trend of how exam scores change with an increase in study hours. The upward slope of the regression line suggests a positive correlation between study hours and exam scores – generally, as the number of study hours per day increases, the exam scores tend to increase as well.

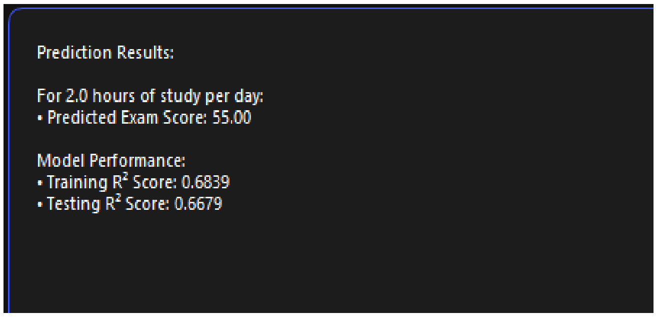
The legend in the upper left corner clarifies that the blue dots represent the "Actual Data" points, while the red line represents the "Regression Line." The x-axis is labeled "Study Hours Per Day," ranging from 0 to 8, and the y-axis is labeled "Exam Score," ranging from 20 to 100. The title of the plot is "Exam Score vs. Study Hours Regression," clearly indicating what the visualization represents.



```
Regression Analysis: Predicting Attendance Percentage from Sleep Hours

Model Performance:
• Training R² Score: 0.0001
• Testing R² Score: -0.0040

Regression Equation:
Attendance Percentage = 0.07 × Sleep Hours + 83.80
```

Attendance Percentage vs. Sleep Hours Regression

This scatter plot shows the relationship between students' sleep hours and their attendance percentage. Green dots represent actual data points, while the red regression line indicates the overall trend. The flatness of the regression line suggests a weak or no significant correlation between sleep duration and attendance.



```
Prediction Results:

For 2.0 hours of study per day:
• Predicted Exam Score: 55.00

Model Performance:
• Training R² Score: 0.6839
• Testing R² Score: 0.6679
```

Prediction Results:

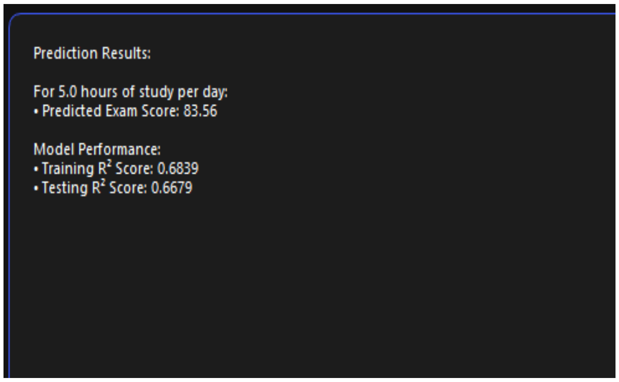
This section presents the results of the exam score prediction based on the input provided in the "Make Prediction" section on the left side of the interface.

- **For 2.0 hours of study per day:**
  - **Predicted Exam Score: 55.00** This indicates that based on the regression model, a student who studies for 2.0 hours per day is predicted to score 55.00 on the exam.

Model Performance:

- **Training R² Score: 0.6839**
  - The R² (R-squared) score for the training data is 0.6839. This value represents the proportion of the variance in the training exam scores that is predictable from the study hours. An R² of 0.6839 suggests that approximately 68.39% of the variation in exam scores in the training data can be explained by the study hours variable.
- **Testing R² Score: 0.6679**
  - The R² score for the testing data is 0.6679. This value indicates the proportion of the variance in the exam scores of unseen data (the test set) that is predictable by the model. An R² of 0.6679 suggests that approximately 66.79% of the variation in exam scores in the testing data can be explained by the study hours variable.

The fact that the training and testing R² scores are relatively close suggests that the model is generalizing reasonably well to new, unseen data.



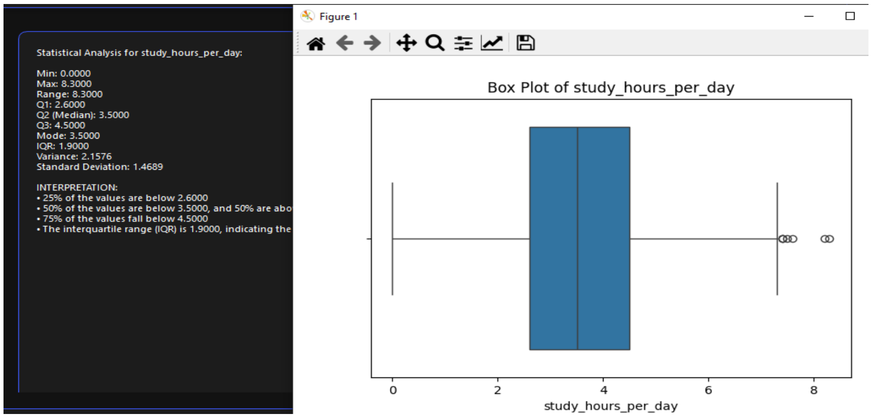
```
Prediction Results:

For 5.0 hours of study per day:
• Predicted Exam Score: 83.56

Model Performance:
• Training R² Score: 0.6839
• Testing R² Score: 0.6679
```

Prediction Results:

This section now displays the results for predicting attendance based on sleep hours, as indicated by the "Predict attendance" selection on the left.

- **For 6.0 hours of sleep per day:**

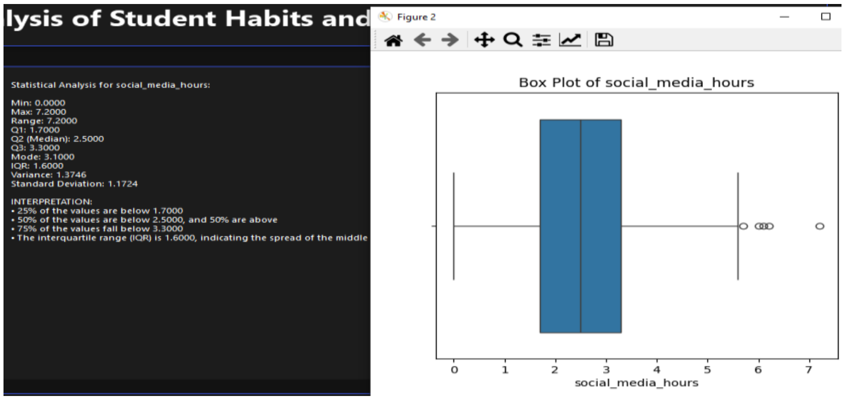
○ **Predicted Attendance Percentage: 84.22%** This indicates that, according to the regression model, a student who sleeps for 6.0 hours per day is predicted to have an attendance percentage of 84.22%.



```
Statistical Analysis for study_hours_per_day:

Min: 0.0000
Max: 8.3000
Range: 8.3000
Q1: 2.6000
Q2 (Median): 3.5000
Q3: 4.5000
Mode: 3.5000
IQR: 1.9000
Variance: 2.1576
Standard Deviation: 1.4689

INTERPRETATION:
• 25% of the values are below 2.6000
• 50% of the values are below 3.5000, and 50% are abo
• 75% of the values fall below 4.5000
• The interquartile range (IQR) is 1.9000, indicating the
```

Box Plot of Study Hours Per Day

The box plot shows that most students study between roughly 2.5 and 4.5 hours daily, with a median of about 3.5 hours. There are a few students who study notably more (around 7.5 to 8 hours), identified as potential outliers. The study hour distribution is slightly skewed towards the lower end.

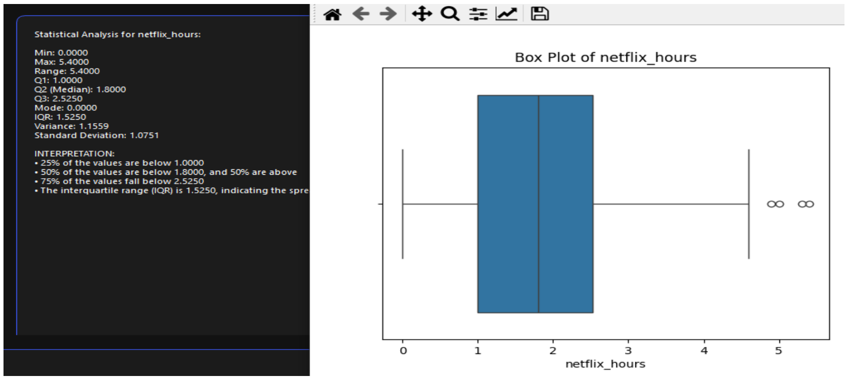


lysis of Student Habits and

```
Statistical Analysis for social_media_hours:

Min: 0.0000
Max: 7.2000
Range: 7.2000
Q1: 1.7000
Q2 (Median): 2.5000
Q3: 3.3000
Mode: 3.1000
IQR: 1.6000
Variance: 1.3746
Standard Deviation: 1.1724

INTERPRETATION:
• 25% of the values are below 1.7000
• 50% of the values are below 2.5000, and 50% are above
• 75% of the values fall below 3.3000
• The interquartile range (IQR) is 1.6000, indicating the spread of the middle
```

Box Plot of Social Media Hours

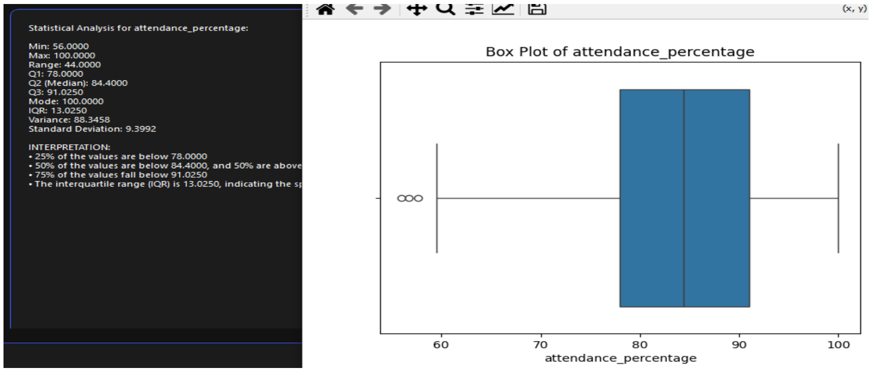
Most students spend between approximately 1.5 and 3 hours on social media daily, with a median of around 2.5 hours. There are several students who spend considerably more time on social media, ranging from about 5.5 to 7.5 hours, and these are identified as potential outliers. The distribution appears somewhat skewed towards the higher end of social media usage.



```
Statistical Analysis for netflix_hours:

Min: 0.0000
Max: 5.4000
Range: 5.4000
Q1: 1.0000
Q2 (Median): 1.8000
Q3: 2.5250
Mode: 0.0000
IQR: 1.5250
Variance: 1.1559
Standard Deviation: 1.0751

INTERPRETATION:
• 25% of the values are below 1.0000
• 50% of the values are below 1.8000, and 50% are above
• 75% of the values fall below 2.5250
• The interquartile range (IQR) is 1.5250, indicating the spre
```

Box Plot of Netflix Hours

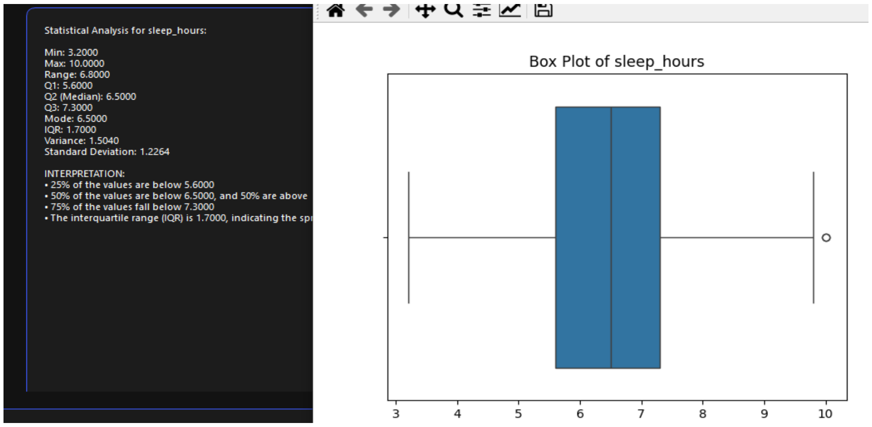
The majority of students watch Netflix for roughly 1 to 2.5 hours daily, with a median of about 1.5 hours. A few students watch considerably more, ranging from around 4.5 to 5.5 hours, and these are shown as potential outliers. The distribution seems slightly skewed towards higher Netflix consumption.



```
Statistical Analysis for attendance_percentage:

Min: 56.0000
Max: 100.0000
Range: 44.0000
Q1: 78.0000
Q2 (Median): 84.4000
Q3: 91.0250
Mode: 100.0000
IQR: 13.0250
Variance: 88.3458
Standard Deviation: 9.3992

INTERPRETATION:
• 25% of the values are below 78.0000
• 50% of the values are below 84.4000, and 50% are above
• 75% of the values fall below 91.0250
• The interquartile range (IQR) is 13.0250, indicating the sp
```

Box Plot of Attendance Percentage

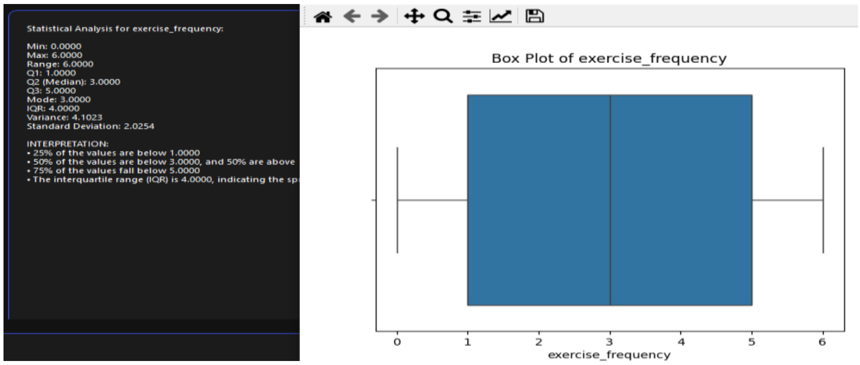
Most students have an attendance percentage between approximately 78% and 92%, with a median around 85%. There are a few students with notably lower attendance, ranging from about 55% to 60%, identified as potential outliers on the lower end. The distribution appears relatively symmetric with a slight lean towards higher attendance.



```
Statistical Analysis for sleep_hours:

Min: 3.2000
Max: 10.0000
Range: 6.8000
Q1: 5.6000
Q2 (Median): 6.5000
Q3: 7.3000
Mode: 6.5000
IQR: 1.7000
Variance: 1.5040
Standard Deviation: 1.2264

INTERPRETATION:
• 25% of the values are below 5.6000
• 50% of the values are below 6.5000, and 50% are above
• 75% of the values fall below 7.3000
• The interquartile range (IQR) is 1.7000, indicating the spr
```

Box Plot of Sleep Hours

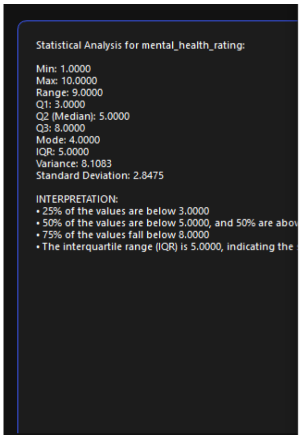
The majority of students sleep between roughly 5.5 and 7 hours, with a median of around 6.25 hours. There's one student who reports a notably higher amount of sleep, around 10 hours, identified as a potential outlier. The distribution of sleep hours appears fairly symmetric



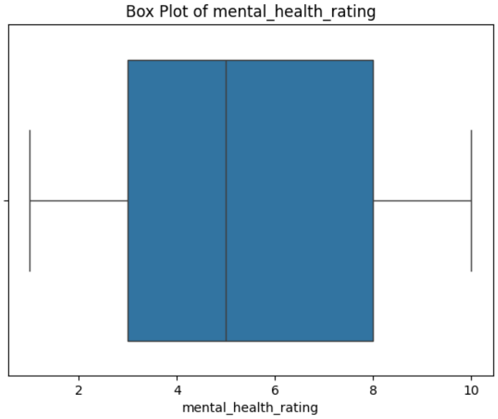
```
Statistical Analysis for exercise_frequency:

Min: 0.0000
Max: 6.0000
Range: 6.0000
Q1: 1.0000
Q2 (Median): 3.0000
Q3: 5.0000
Mode: 3.0000
IQR: 4.0000
Variance: 4.1023
Standard Deviation: 2.0254

INTERPRETATION:
• 25% of the values are below 1.0000
• 50% of the values are below 3.0000, and 50% are above
• 75% of the values fall below 5.0000
• The interquartile range (IQR) is 4.0000, indicating the sp
```

Box Plot of Exercise Frequency

Most students exercise between approximately 1 and 5 times per week, with a median of around 3 times. The distribution of exercise frequency appears relatively symmetrical, with a similar spread below and above the median. There don't appear to be any significant outliers in this data.

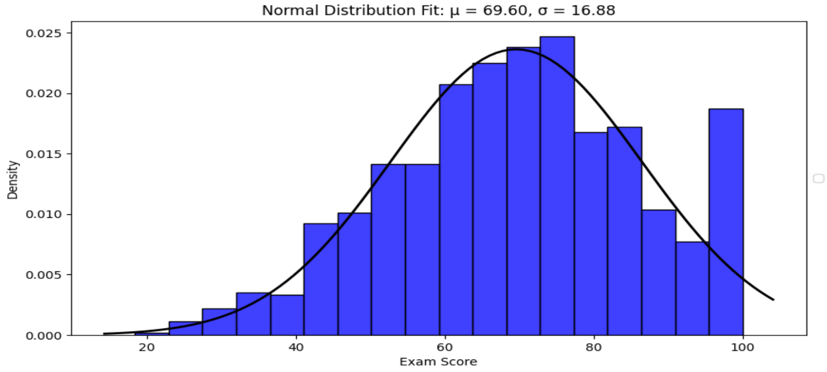


```
Statistical Analysis for mental_health_rating:

Min: 1.0000
Max: 10.0000
Range: 9.0000
Q1: 3.0000
Q2 (Median): 5.0000
Q3: 8.0000
Mode: 4.0000
IQR: 5.0000
Variance: 8.1083
Standard Deviation: 2.8475

INTERPRETATION:
• 25% of the values are below 3.0000
• 50% of the values are below 5.0000, and 50% are abov
• 75% of the values fall below 8.0000
• The interquartile range (IQR) is 5.0000, indicating the
```



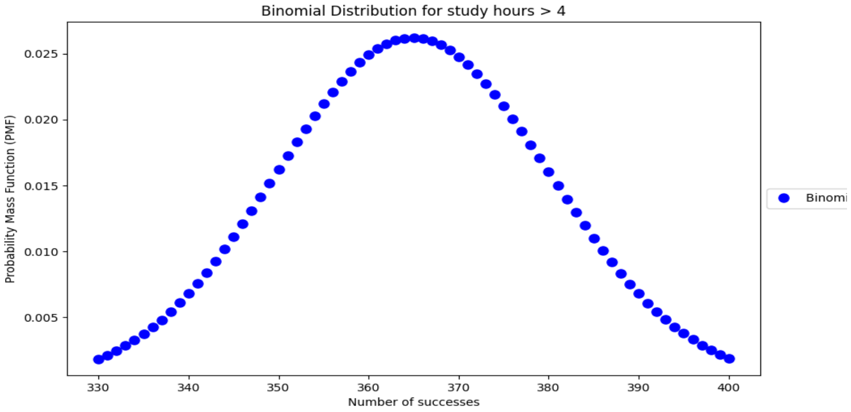
Box Plot of Mental Health Rating

The majority of students have a mental health rating between approximately 4 and 8, with a median rating of around 6. The distribution of mental health ratings appears relatively symmetrical, suggesting a balanced spread of scores. There don't seem to be any significant outliers in the reported mental health ratings.



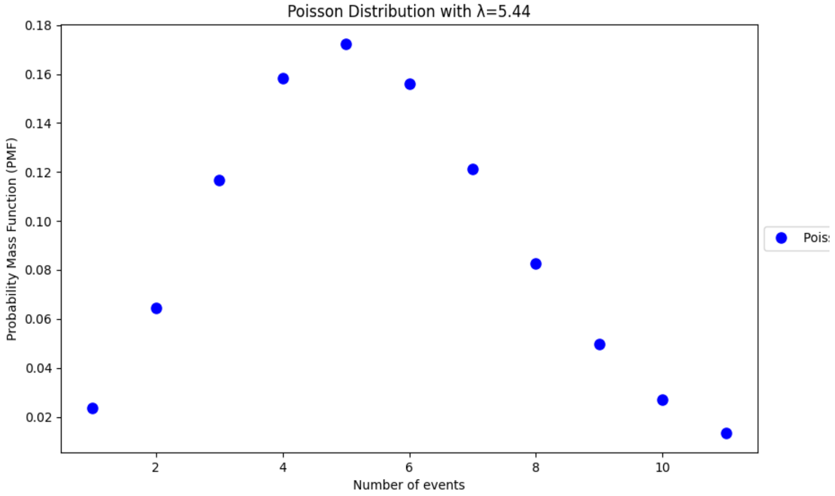
Histogram of Exam Scores with Normal Distribution

This histogram shows the distribution of exam scores, with blue bars representing score frequencies. The x-axis shows scores, and the y-axis shows density. A black curve overlays a normal distribution with mean μ = 69.60 and standard deviation σ = 16.88. The distribution is roughly bell-shaped with a slight positive skew, suggesting the normal curve is a reasonable but not perfect fit.



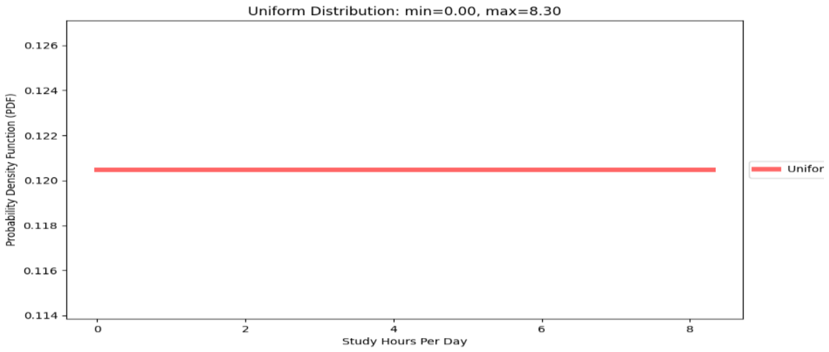
Binomial Distribution for study hours > 4

This plot shows a binomial distribution for the condition "study hours > 4." The x-axis represents the number of successes (approx. 330–400), and the y-axis shows the probability (PMF). Blue circles represent the PMF, peaking around 365 successes. The distribution is symmetric and bell-shaped, typical for binomial distributions with many trials. The exact meaning of "success" is not specified in the plot.



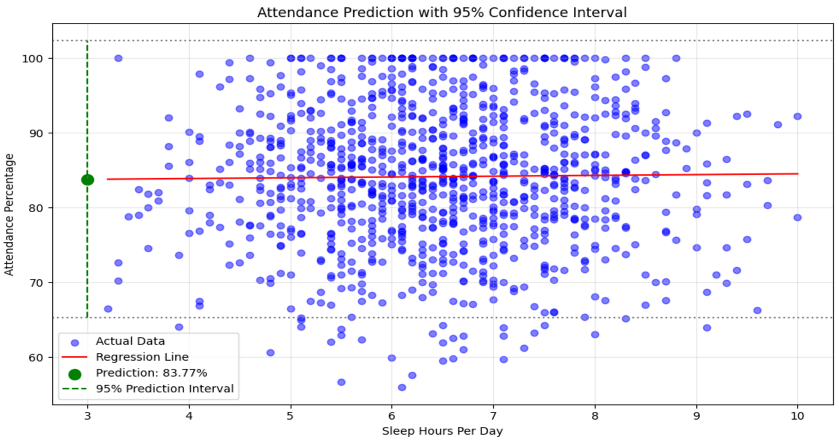
Poisson Distribution with λ=5.44

This plot shows a Poisson distribution, modeling the probability of a certain number of events occurring within a fixed interval, given an average rate (λ) of 5.44 events. The blue dots represent the probability of observing a specific "Number of events" (x-axis), with the highest probability around 5 events. The probability decreases as the number of events deviates from this average rate.



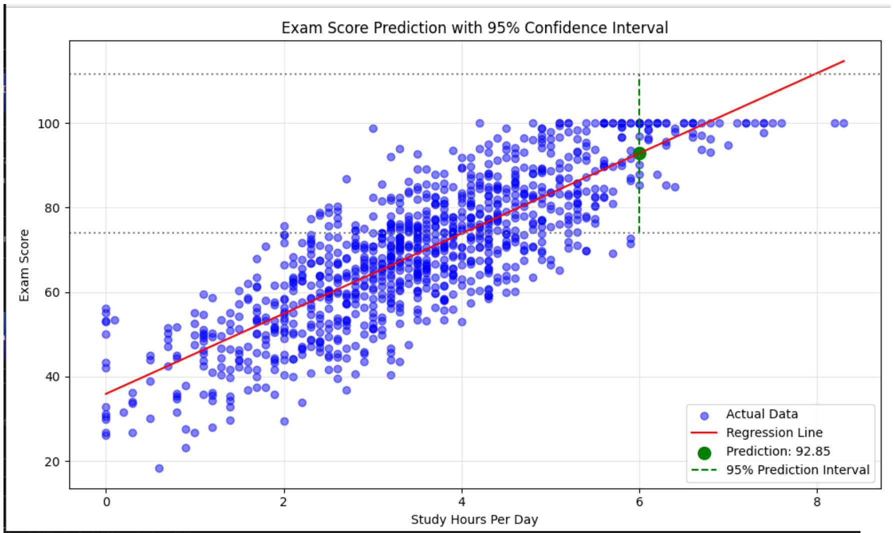
Uniform Distribution: min=0.00, max=8.30

This plot shows a uniform distribution for "Study Hours Per Day," ranging from a minimum of 0.00 to a maximum of 8.30. The red horizontal line indicates that every value within this range has an equal probability density of approximately 0.12.



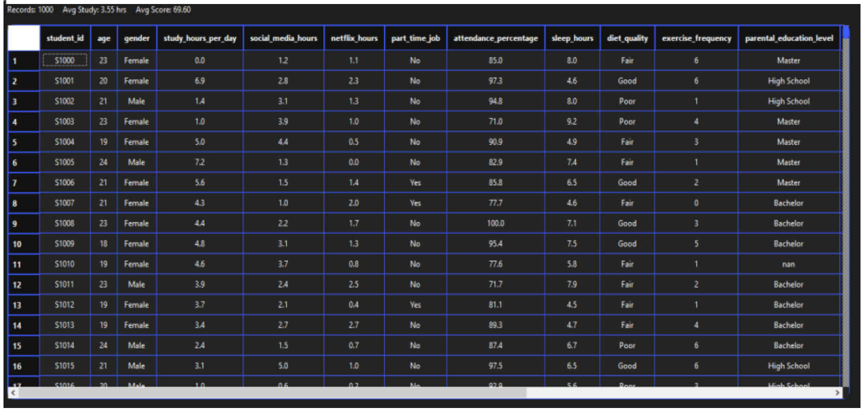
Attendance Prediction vs. Sleep Hours with 95% Confidence Interval

The plot shows a weak relationship between sleep hours and attendance. For about 3 hours of sleep, the predicted attendance is 83.77% (green circle), but the wide 95% prediction interval (dashed green lines) indicates high uncertainty in this prediction. The flat red line further suggests that sleep hours aren't a strong predictor of attendance in this model.



Exam Score Prediction vs. Study Hours with 95% Confidence Interval

The plot shows a positive relationship between study hours and exam scores. For approximately 6 hours of study per day (vertical dashed green line), the predicted exam score is 92.85 (large green circle). The 95% prediction interval (horizontal dashed green lines) around this prediction indicates the range within which we can be 95% confident a new student's score will fall for that study time. The upward sloping red regression line confirms the general trend of higher study hours leading to higher predicted exam scores.

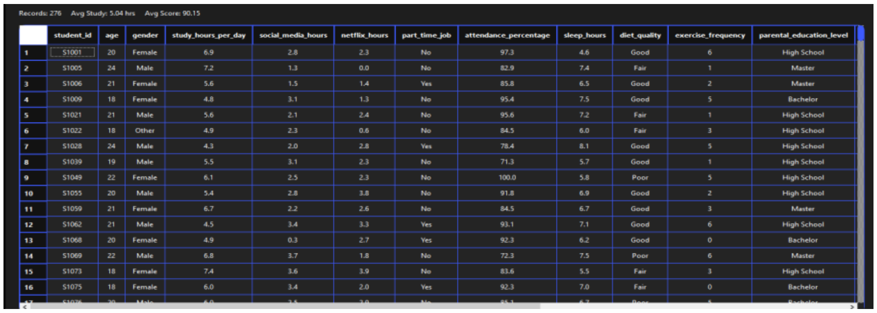


Records: 1000 Avg Study: 3.55 hrs Avg Score: 69.60

	student_id	age	gender	study_hours_per_day	social_media_hours	netflix_hours	part_time_job	attendance_percentage	sleep_hours	diet_quality	exercise_frequency	parental_education_level
1	S1000	23	Female	0.0	1.2	1.1	No	85.0	8.0	Fair	6	Master
2	S1001	20	Female	6.9	2.8	2.3	No	97.3	4.6	Good	6	High School
3	S1002	21	Male	1.4	3.1	1.3	No	94.8	8.0	Poor	1	High School
4	S1003	23	Female	1.0	3.9	1.0	No	71.0	9.2	Poor	4	Master
5	S1004	19	Female	5.0	4.4	0.5	No	90.9	4.9	Fair	3	Master
6	S1005	24	Male	7.2	1.3	0.0	No	82.9	7.4	Fair	1	Master
7	S1006	21	Female	5.6	1.5	1.4	Yes	85.8	6.5	Good	2	Master
8	S1007	21	Female	4.3	1.0	2.0	Yes	77.7	4.6	Fair	0	Bachelor
9	S1008	23	Female	4.4	2.2	1.7	No	100.0	7.1	Good	3	Bachelor
10	S1009	18	Female	4.8	3.1	1.3	No	95.4	7.5	Good	5	Bachelor
11	S1010	19	Female	4.6	3.7	0.8	No	77.6	5.8	Fair	1	nan
12	S1011	23	Male	3.9	2.4	2.5	No	71.7	7.9	Fair	2	Bachelor
13	S1012	19	Female	3.7	2.1	0.4	Yes	81.1	4.5	Fair	1	Bachelor
14	S1013	19	Female	3.4	2.7	2.7	No	89.3	4.7	Fair	4	Bachelor
15	S1014	24	Male	2.4	1.5	0.7	No	87.4	6.7	Poor	6	Bachelor
16	S1015	21	Male	3.1	5.0	1.0	No	97.5	6.5	Good	6	High School
17	S1016	20	Male	1.0	0.6	0.2	No	92.9	5.6	Poor	3	High School

Data Summary for All Data:

Total Records: 1000,Average Study Hours: 3.55,Average Sleep Hours: 6.47,Average Exam Score: 69.60, Highest Exam Score: 100.00,Lowest Exam Score: 18.40. The data table has been displayed in a separate window.

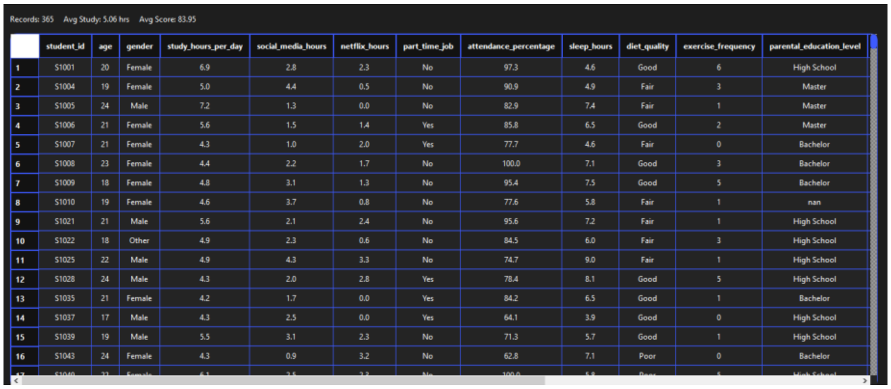


Records: 276 Avg Study: 5.04 hrs Avg Score: 90.15

	student_id	age	gender	study_hours_per_day	social_media_hours	netflix_hours	part_time_job	attendance_percentage	sleep_hours	diet_quality	exercise_frequency	parental_education_level
1	S1001	20	Female	6.9	2.8	2.3	No	97.3	4.6	Good	6	High School
2	S1005	24	Male	7.2	1.3	0.0	No	82.9	7.4	Fair	1	Master
3	S1006	21	Female	5.6	1.5	1.4	Yes	85.8	6.5	Good	2	Master
4	S1009	18	Female	4.8	3.1	1.3	No	95.4	7.5	Good	5	Bachelor
5	S1021	21	Male	5.6	2.1	2.4	No	95.6	7.2	Fair	1	High School
6	S1022	18	Other	4.9	2.3	0.6	No	84.5	6.0	Fair	3	High School
7	S1028	24	Male	4.3	2.0	2.8	Yes	78.4	8.1	Good	5	High School
8	S1039	19	Male	5.5	3.1	2.3	No	71.3	5.7	Good	1	High School
9	S1049	22	Female	6.1	2.5	2.3	No	100.0	5.8	Poor	5	High School
10	S1055	20	Male	5.4	2.8	3.8	No	91.8	6.9	Good	2	High School
11	S1059	21	Female	6.7	2.2	2.6	No	84.5	6.7	Good	3	Master
12	S1062	21	Male	4.5	3.4	3.3	Yes	93.1	7.1	Good	6	High School
13	S1068	20	Female	4.9	0.3	2.7	Yes	92.3	6.2	Good	0	Bachelor
14	S1069	22	Male	6.8	3.7	1.8	No	72.3	7.5	Poor	6	Master
15	S1073	18	Female	7.4	3.6	3.9	No	83.6	5.5	Fair	3	High School
16	S1075	18	Female	6.0	3.4	2.0	Yes	92.3	7.0	Fair	0	Bachelor

Data Summary for High Performers (Exam > 80):

Total Records: 276,Average Study Hours: 5.04,Average Sleep Hours: 6.68,Average Exam Score: 90.15,Highest Exam Score: 100.00,Lowest Exam Score: 80.20.The data table has been displayed in a separate window.



Records: 365 Avg Study: 5.06 hrs Avg Score: 83.95

	student_id	age	gender	study_hours_per_day	social_media_hours	netflix_hours	part_time_job	attendance_percentage	sleep_hours	diet_quality	exercise_frequency	parental_education_level
1	S1001	20	Female	6.9	2.8	2.3	No	97.3	4.6	Good	6	High School
2	S1004	19	Female	5.0	4.4	0.5	No	90.9	4.9	Fair	3	Master
3	S1005	24	Male	7.2	1.3	0.0	No	82.9	7.4	Fair	1	Master
4	S1006	21	Female	5.6	1.5	1.4	Yes	85.8	6.5	Good	2	Master
5	S1007	21	Female	4.3	1.0	2.0	Yes	77.7	4.6	Fair	0	Bachelor
6	S1008	23	Female	4.4	2.2	1.7	No	100.0	7.1	Good	3	Bachelor
7	S1009	18	Female	4.8	3.1	1.3	No	95.4	7.5	Good	5	Bachelor
8	S1010	19	Female	4.6	3.7	0.8	No	77.6	5.8	Fair	1	nan
9	S1021	21	Male	5.6	2.1	2.4	No	95.6	7.2	Fair	1	High School
10	S1022	18	Other	4.9	2.3	0.6	No	84.5	6.0	Fair	3	High School
11	S1025	22	Male	4.9	4.3	3.3	No	74.7	9.0	Fair	1	High School
12	S1028	24	Male	4.3	2.0	2.8	Yes	78.4	8.1	Good	5	High School
13	S1035	21	Female	4.2	1.7	0.0	Yes	84.2	6.5	Good	1	Bachelor
14	S1037	17	Male	4.3	2.5	0.0	Yes	64.1	3.9	Good	0	High School
15	S1039	19	Male	5.5	3.1	2.3	No	71.3	5.7	Good	1	High School
16	S1043	24	Female	4.3	0.9	3.2	No	62.8	7.1	Poor	0	Bachelor

Data Summary for Study Hours > 4:

Total Records: 365,Average Study Hours: 5.06,Average Sleep Hours: 6.41,Average Exam Score: 83.95,Highest Exam Score: 100.00,Lowest Exam Score: 57.40.The data table has been displayed in a separate window.

5.Code

```
import sys

import pandas as pd
import numpy as np
import seaborn as sns
from scipy.stats import norm, poisson, uniform, binom
import matplotlib.pyplot as plt
from PyQt5.QtWidgets import QApplication, QTableWidget,
QTableWidgetItem, QDialog, QVBoxLayout, QHeaderView,
QPushButton, QLabel, QHBoxLayout, QSizePolicy
from PyQt5.QtCore import QTimer, Qt
from PyQt5.QtGui import QFont, QColor
from modern_ui import ModernMainWindow
class StudentHabitsAnalysisApp(ModernMainWindow):
  def __init__(self):
    super().__init__()
    # Load the dataset
    self.data = pd.read_csv('student_habits_performance.csv')
    # Connect button signals to slots
    self.connect_signals()
    # Set window title
    self.setWindowTitle("Student Habits and Performance Analysis")
  def connect_signals(self):
    # Connect all button click events to their respective functions
    self.SearchPie.clicked.connect(self.get_pie_chart)
    self.SearchPie_distribution.clicked.connect(self.get_distribution)
    self.SearchPie_EDA.clicked.connect(self.get_eda)
    self.SearchPie_Regression.clicked.connect(self.get_regression)

self.SearchPie_Regression_predict.clicked.connect(self.get_regressi
on_predict)
    self.SearchPie_Confidence.clicked.connect(self.get_Confidence)
    # Connect the data table button
    self.data_table_button.clicked.connect(self.get_data_table)
  def show_loading_and_execute(self, func):
    """Helper method to show loading animation and execute a
function"""
    self.show_loading()
    # Use a timer to delay execution slightly to show the loading
animation
    QTimer.singleShot(500, lambda:
self.execute_with_loading(func))
  def execute_with_loading(self, func):
    """Execute the function and hide loading when done"""
    try:
      func()
    finally:
      self.hide_loading()
  def get_pie_chart(self):
    """Generate a pie chart for the selected categorical variable"""
    self.show_loading_and_execute(self._get_pie_chart)
  def _get_pie_chart(self):
    # Get the selected variable
    var = self.comboPie.currentText()
    # Calculate value counts
    value_counts = self.data[var].value_counts()
    # Select top categories and group the rest into "Other"
    top_categories = value_counts.head(10)
    other_sum = value_counts[10:].sum() if len(value_counts) > 10
else 0
    if other_sum != 0:
      top_categories['Other'] = other_sum
    # Calculate percentages
    total = top_categories.sum()
    percentages = (top_categories / total * 100).round(1)
    # Create labels with percentages
    labels = [f'{name}: {percent}%' for name, percent in
zip(top_categories.index, percentages)]
    # Create a pie chart
    plt.figure(figsize=(13, 6))
    plt.pie(top_categories, labels=labels, autopct='%1.1f%%',
startangle=90,
        colors=plt.cm.Paired(range(len(top_categories))))
    plt.title(f'Pie Chart of {var} with Percentages')
```

```
        plt.legend(loc='center left', bbox_to_anchor=(1, 0.5))
        plt.show()
        # Calculate the mode
        mode_value = self.data[var].mode()[0]
        # Create description text
        description = (
            f"Detailed Percentages for Each Category in {var}:\n" +
            "\n".join([f'{name}: {percent}%' for name, percent in zip(top_categories.index, percentages)]) +
            f"\n\nThe mode for {var} is: {mode_value}\nWhich means that in {var}\n {mode_value} is most popular"
        )
        # Update the output text
        self.output_text.setText(description)
    def get_distribution(self):
        """Generate a probability distribution for the selected type"""
        self.show_loading_and_execute(self._get_distribution)
    def _get_distribution(self):
        # Get the selected distribution type
        dist_type = self.comboPie_distribution.currentText()
        if dist_type == "Binomial Distribution":
            n = len(self.data)  # number of trials
            p = np.mean(self.data['study_hours_per_day'] > 4)  # probability of success (studying more than 4 hours)
            if 0 < p < 1:
                binomial_dist = binom(n=n, p=p)
                x = np.arange(binom.ppf(0.01, n, p), binom.ppf(0.99, n, p))
                plt.figure(figsize=(10, 6))
                plt.plot(x, binomial_dist.pmf(x), 'bo', ms=8, label='Binomial PMF')
                plt.title('Binomial Distribution for study hours > 4')
                plt.xlabel('Number of successes')
                plt.ylabel('Probability Mass Function (PMF)')
                plt.legend(loc='center left', bbox_to_anchor=(1, 0.5))
                plt.show()
                description = f"Binomial Distribution with n={n} and p={p:.4f}"
                self.output_text.setText(description)
            else:
                self.output_text.setText(f"Error: Probability 'p' must be between 0 and 1 but got: {p}")
        elif dist_type == "Poisson Distribution":
            data_sample = self.data['mental_health_rating']  # Using mental health rating for Poisson
            rate = np.mean(data_sample)  # The rate parameter (lambda) for Poisson
            poisson_dist = poisson(rate)
            x = np.arange(poisson.ppf(0.01, rate), poisson.ppf(0.99, rate))
            plt.figure(figsize=(10, 6))
    plt.plot(x, poisson_dist.pmf(x), 'bo', ms=8, label='Poisson PMF')
            plt.title(f'Poisson Distribution with λ={rate:.2f}')
            plt.xlabel('Number of events')
            plt.ylabel('Probability Mass Function (PMF)')
            plt.legend(loc='center left', bbox_to_anchor=(1, 0.5))
            plt.show()
            description = f"Poisson Distribution with λ (rate) = {rate:.2f}"
            self.output_text.setText(description)
        elif dist_type == "Normal Distribution":
            data_sample = self.data['exam_score']
            mu, std = norm.fit(data_sample)
            plt.figure(figsize=(10, 6))
            sns.histplot(data_sample, kde=False, color='blue', stat="density")
            xmin, xmax = plt.xlim()
            x = np.linspace(xmin, xmax, 100)
            p = norm.pdf(x, mu, std)
            plt.plot(x, p, 'k', linewidth=2)
            title = f"Normal Distribution Fit: μ = {mu:.2f}, σ = {std:.2f}"
            plt.title(title)
            plt.xlabel('Exam Score')
            plt.ylabel('Density')
            plt.legend(loc='center left', bbox_to_anchor=(1, 0.5))
            plt.show()
            description = f"Normal Distribution with μ (mean) = {mu:.2f} and σ (std) = {std:.2f}"
            self.output_text.setText(description)
        elif dist_type == "Uniform Distribution":
            data_sample = self.data['study_hours_per_day']
            min_val = min(data_sample)
            max_val = max(data_sample)
            width = max_val - min_val
            uniform_dist = uniform(loc=min_val, scale=width)
            x = np.linspace(min_val, max_val, 100)
            plt.figure(figsize=(10, 6))
            plt.plot(x, uniform_dist.pdf(x), 'r-', lw=5, alpha=0.6, label='Uniform PDF')
            plt.title(f'Uniform Distribution: min={min_val:.2f}, max={max_val:.2f}')
            plt.xlabel('Study Hours Per Day')
            plt.ylabel('Probability Density Function (PDF)')
            plt.legend(loc='center left', bbox_to_anchor=(1, 0.5))
            plt.show()
            description = f"Uniform Distribution with min={min_val:.2f} and max={max_val:.2f}"
            self.output_text.setText(description)
    def get_eda(self):
        """Perform exploratory data analysis on the selected variable"""
        self.show_loading_and_execute(self._get_eda)
    def _get_eda(self):
        # Get the selected variable
        var = self.comboPie_EDA.currentText()
        # Calculate descriptive statistics
        min_val = self.data[var].min()
        max_val = self.data[var].max()
        range_val = max_val - min_val
        quartiles = self.data[var].quantile([0.25, 0.5, 0.75])
        mode = self.data[var].mode()[0]
        iqr = quartiles[0.75] - quartiles[0.25]
        variance = self.data[var].var()
        std_dev = self.data[var].std()
        # Store results in dictionary
        stats_dict = {
            'Min': min_val,
            'Max': max_val,
            'Range': range_val,
            'Q1': quartiles[0.25],
            'Q2 (Median)': quartiles[0.5],
            'Q3': quartiles[0.75],
            'Mode': mode,
            'IQR': iqr,
            'Variance': variance,
            'Standard Deviation': std_dev
        }
        # Generate box plot
        plt.figure(figsize=(10, 5))
        sns.boxplot(x=self.data[var])
        plt.title(f'Box Plot of {var}')
        plt.xlabel(var)
        plt.show()
        # Create description text
        description = (
            f"Statistical Analysis for {var}:\n\n" +
            "\n".join([f"{key}: {value:.4f}" for key, value in stats_dict.items()]) +
            "\n\nINTERPRETATION:" +
            f"\n• 25% of the values are below {stats_dict['Q1']:.4f}" +
            f"\n• 50% of the values are below {stats_dict['Q2 (Median)']:.4f}, and 50% are above" +
            f"\n• 75% of the values fall below {stats_dict['Q3']:.4f}" +
            f"\n• The interquartile range (IQR) is {stats_dict['IQR']:.4f}, indicating the spread of the middle 50% of the data"
        )
        # Update the output text
        self.output_text.setText(description)
    def get_regression(self):
        """Generate a  regression model for the selected prediction type"""
        self.show_loading_and_execute(self._get_regression)
    def _get_regression(self):
        from sklearn.model_selection import train_test_split
        from sklearn.linear_model import LinearRegression
        # Get the selected prediction type
        pred_type = self.comboPie_Regression.currentText()
        if pred_type == "Predict exam score":
            X = self.data[['study_hours_per_day']]
            y = self.data['exam_score']
            # Split the data
            X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state=42)
            # Create and train the model
            model = LinearRegression()
            model.fit(X_train, y_train)
            # Generate prediction points for plotting
            hours_points = np.linspace(self.data['study_hours_per_day'].min(),
                                       self.data['study_hours_per_day'].max(), 100).reshape(-1, 1)
            predicted_values = model.predict(hours_points)
            # Plot the results
            plt.figure(figsize=(12, 6))
            sns.scatterplot(x=X.squeeze(), y=y, color='blue', alpha=0.5, label='Actual Data')
            plt.plot(hours_points, predicted_values, color='red', linewidth=2, label='Regression Line')
            plt.title('Exam Score vs. Study Hours Regression')
            plt.xlabel('Study Hours Per Day')
            plt.ylabel('Exam Score')
            plt.legend()
            plt.show()
            # Calculate model performance
            train_score = model.score(X_train, y_train)
            test_score = model.score(X_test, y_test)
            # Create description text
            description = (
                f"Regression Analysis: Predicting Exam Score from Study Hours\n\n" +
                f"Model  Performance:\n" +
                f"• Training R² Score: {train_score:.4f}\n" +
                f"• Testing R² Score: {test_score:.4f}\n\n" +
                f"Regression Equation:\n" +
                f"Exam Score = {model.coef_[0]:.2f} × Study Hours + {model.intercept_:.2f}\n\n" +
                f"Interpretation:\n" +
                f"• For each additional hour of study, exam score changes by {model.coef_[0]:.2f} points\n" +
                f"• The model explains {train_score*100:.1f}% of the variance in exam scores"
            )
            # Update the output text
            self.output_text.setText(description)
        elif pred_type == "Predict attendance":
            X = self.data[['sleep_hours']]
            y = self.data['attendance_percentage']
                # Split the data
            X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state=42)
            # Create and train the model
            model = LinearRegression()
            model.fit(X_train, y_train)
            # Generate prediction points for plotting
            sleep_points = np.linspace(self.data['sleep_hours'].min(),
                                  self.data['sleep_hours'].max(), 100).reshape(-1, 1)
            predicted_values = model.predict(sleep_points)
            # Plot the results
            plt.figure(figsize=(12, 6))
            sns.scatterplot(x=X.squeeze(), y=y, color='green', alpha=0.5, label='Actual Data')
            plt.plot(sleep_points, predicted_values, color='red', linewidth=2, label='Regression Line')
            plt.title('Attendance Percentage vs. Sleep Hours Regression')
            plt.xlabel('Sleep Hours')
            plt.ylabel('Attendance Percentage')
            plt.legend()
            plt.show()
            # Calculate model performance
            train_score = model.score(X_train, y_train)
            test_score = model.score(X_test, y_test)
            # Create description text
            description = (
                f"Regression Analysis: Predicting Attendance Percentage from Sleep Hours\n\n" +
                f"Model  Performance:\n" +
                f"• Training R² Score: {train_score:.4f}\n" +
                f"• Testing R² Score: {test_score:.4f}\n\n" +
                f"Regression Equation:\n" +
                f"Attendance Percentage = {model.coef_[0]:.2f} × Sleep Hours + {model.intercept_:.2f}\n\n" +
                f"Interpretation:\n" +
                f"• For each additional hour of sleep, attendance percentage changes by {model.coef_[0]:.2f}%\n" +
                f"• The model explains {train_score*100:.1f}% of the variance in attendance percentage"
            )
            # Update the output text
            self.output_text.setText(description)
    def get_regression_predict(self):
        """Make  a prediction using the regression model"""
        self.show_loading_and_execute(self._get_regression_predict)
    def _get_regression_predict(self):
        from sklearn.model_selection import train_test_split
        from sklearn.linear_model import LinearRegression
        # Get the selected prediction type and input value
        pred_type = self.comboPie_Regression_predict.currentText()
        input_value = self.spinBox_predict.value()
        # Predicting Exam Score based on Study Hours
        X_exam = self.data[['study_hours_per_day']]
        y_exam = self.data['exam_score']
        # Splitting the data for the exam score model
        X_train_e, X_test_e, y_train_e, y_test_e = train_test_split(X_exam, y_exam, test_size=0.2, random_state=42)
        # Create and train the exam score model
        model_exam = LinearRegression()
        model_exam.fit(X_train_e, y_train_e)
        # Predicting Attendance based on Sleep Hours
        X_attend = self.data[['sleep_hours']]
        y_attend = self.data['attendance_percentage']
        # Splitting the data for the attendance model
        X_train_a, X_test_a, y_train_a, y_test_a =
train_test_split(X_attend, y_attend, test_size=0.2, random_state=42)
        # Create and train the attendance model
        model_attend = LinearRegression()
        model_attend.fit(X_train_a, y_train_a)
        # Use the models to predict based on the input value
        input_point = [[input_value]]
        if pred_type == "Predict exam score":
            predicted_exam = model_exam.predict(input_point)
            description = (
                f"Prediction Results:\n\n" +
                f"For {input_value} hours of study per day:\n" +
                f"• Predicted Exam Score: {predicted_exam[0]:.2f}\n\n" +
                f"Model Performance:\n" +
                f"• Training R^2 Score: {model_exam.score(X_train_e,
y_train_e):.4f}\n" +
                f"• Testing R^2 Score: {model_exam.score(X_test_e,
y_test_e):.4f}"
            )
            self.output_text.setText(description)
        elif pred_type == "Predict attendance":
            predicted_attend = model_attend.predict(input_point)
            description = (
                f"Prediction Results:\n\n" +
                f"For {input_value} hours of sleep per day:\n" +
                f"• Predicted Attendance Percentage:
{predicted_attend[0]:.2f}%\n\n" +
                f"Model Performance:\n" +
                f"• Training R^2 Score: {model_attend.score(X_train_a,
y_train_a):.4f}\n" +
                f"• Testing R^2 Score: {model_attend.score(X_test_a,
y_test_a):.4f}"
            )
            self.output_text.setText(description)
    def get_Confidence(self):
        """Calculate and display confidence intervals for predictions"""
        self.show_loading_and_execute(self._get_Confidence)
    def _get_Confidence(self):
        import statsmodels.api as sm
        from statsmodels.formula.api import ols
        import scipy.stats as stats
        # Get the selected confidence interval type
        interval_type = self.comboPie_Confidence.currentText()
        if interval_type == "Exam Score vs Study Hours":
            # Get study hours from user input
            study_hours = self.spinBox_km.value()
            # Get the data for regression
            x = self.data['study_hours_per_day']
            y = self.data['exam_score']
            # Manually calculate regression parameters
            slope, intercept, r_value, p_value, std_err = stats.linregress(x, y)
            # Calculate predicted value for the input study hours
            predicted_score = intercept + slope * study_hours
            # Calculate the standard error of the regression
            n = len(x)
            y_pred = intercept + slope * x
            residuals = y - y_pred
            residual_std = np.sqrt(np.sum(residuals**2) / (n-2))
            # Calculate prediction interval
            x_mean = np.mean(x)
            x_std = np.std(x)
            # Standard error of prediction
            se_pred = residual_std * np.sqrt(1 + 1/n + (study_hours -
x_mean)**2 / ((n-1) * x_std**2))
            # t-value for 95% confidence
            t_value = stats.t.ppf(0.975, n-2)
            # Lower and upper bounds of prediction interval
            lower = predicted_score - t_value * se_pred
            upper = predicted_score + t_value * se_pred
            description = (
                f"Predicted Exam Score for {study_hours:,.1f} hours of study
per day:\n"
                f"    {predicted_score:.2f} points\n\n"
                f"95%  prediction interval for a student studying
{study_hours:,.1f} hours per day: \n"
                f"    {lower:.2f} to {upper:.2f} points\n\n"
                f"Interpretation:\n"
                f"• With 95% confidence, a student studying
{study_hours:,.1f} hours per day will score\n"
                f"  between {lower:.2f} and {upper:.2f} on the exam.\n"
                f"• The correlation between study hours and exam score is:
{r_value:.4f}\n"
                f"• The regression equation is: Score = {intercept:.2f} +
{slope:.2f} × Study Hours"
            )
            # Plot the confidence interval
            plt.figure(figsize=(10, 6))
            # Plot the actual data points
            plt.scatter(x, y, alpha=0.5, color='blue', label='Actual Data')
            # Generate x values for the regression line
            x_line = np.linspace(min(x), max(x), 100)
            # Calculate y values for the regression line
            y_line = intercept + slope * x_line
            # Plot the regression line
            plt.plot(x_line, y_line, color='red', label='Regression Line')
            # Highlight the prediction point
            plt.scatter([study_hours], [predicted_score], color='green',
s=100,
                    label=f'Prediction: {predicted_score:.2f}')
            # Add a vertical interval line for the prediction
            plt.vlines(x=study_hours, ymin=lower, ymax=upper,
                    colors='green', linestyles='dashed', label='95% Prediction
Interval')
            plt.axhline(y=lower, color='gray', linestyle='dotted')
            plt.axhline(y=upper, color='gray', linestyle='dotted')
            plt.xlabel('Study Hours Per Day')
            plt.ylabel('Exam Score')
            plt.title('Exam Score Prediction with 95% Confidence Interval')
            plt.legend()
            plt.grid(True, alpha=0.3)
            plt.tight_layout()
            plt.show()
        elif interval_type == "Attendance vs Sleep Hours":
            # Get sleep hours from user input
            sleep_hours = self.spinBox_km.value()
            # Get the data for regression
            x = self.data['sleep_hours']
            y = self.data['attendance_percentage']
            # Manually calculate regression parameters
            slope, intercept, r_value, p_value, std_err = stats.linregress(x, y)
            # Calculate predicted value for the input sleep hours
            predicted_attendance = intercept + slope * sleep_hours
            # Calculate the standard error of the regression
            n = len(x)
            y_pred = intercept + slope * x
            residuals = y - y_pred
            residual_std = np.sqrt(np.sum(residuals**2) / (n-2))
           # Calculate prediction interval
            x_mean = np.mean(x)
            x_std = np.std(x)
            # Standard error of prediction
            se_pred = residual_std * np.sqrt(1 + 1/n + (sleep_hours -
x_mean)**2 / ((n-1) * x_std**2))
            # t-value for 95% confidence
            t_value = stats.t.ppf(0.975, n-2)
            # Lower and upper bounds of prediction interval
            lower = predicted_attendance - t_value * se_pred
            upper = predicted_attendance + t_value * se_pred
            description = (
                f"Predicted Attendance Percentage for {sleep_hours:,.1f}
hours of sleep per day:\n"
                f"    {predicted_attendance:.2f}%\n\n"
                f"95%  prediction interval for a student sleeping
{sleep_hours:,.1f} hours per day: \n"
                f"    {lower:.2f}% to {upper:.2f}%\n\n"
                f"Interpretation:\n"
                f"• With 95% confidence, a student sleeping
{sleep_hours:,.1f} hours per day will have\n"
                f"  between {lower:.2f}% and {upper:.2f}% attendance.\n"
                f"• The correlation between sleep hours and attendance is:
{r_value:.4f}\n"
                f"• The regression equation is: Attendance = {intercept:.2f} +
{slope:.2f} × Sleep Hours"
            )
            # Plot the confidence interval
            plt.figure(figsize=(10, 6))
            # Plot the actual data points
            plt.scatter(x, y, alpha=0.5, color='blue', label='Actual Data')
            # Generate x values for the regression line
            x_line = np.linspace(min(x), max(x), 100)
            # Calculate y values for the regression line
            y_line = intercept + slope * x_line
            # Plot the regression line
            plt.plot(x_line, y_line, color='red', label='Regression Line')
            # Highlight the prediction point
            plt.scatter([sleep_hours], [predicted_attendance],
color='green', s=100,
                        label=f'Prediction: {predicted_attendance:.2f}%')
            # Add a vertical interval line for the prediction
            plt.vlines(x=sleep_hours, ymin=lower, ymax=upper,
                      colors='green', linestyles='dashed', label='95% Prediction
Interval')
            plt.axhline(y=lower, color='gray', linestyle='dotted')
            plt.axhline(y=upper, color='gray', linestyle='dotted')
            plt.xlabel('Sleep Hours Per Day')
            plt.ylabel('Attendance Percentage')
            plt.title('Attendance Prediction with 95% Confidence Interval')
            plt.legend()
            plt.grid(True, alpha=0.3)
            plt.tight_layout()
            plt.show()
        self.output_text.setText(description)
    def get_data_table(self):
        """Display the data in a tabular format based on the selected
filter"""
        self.show_loading_and_execute(self._get_data_table)
    def _get_data_table(self):
        # Get the selected filter
        filter_option = self.data_table_filter_combo.currentText()
        # Apply filter to the data
        if filter_option == "All Data":
            filtered_data = self.data
        elif filter_option == "High Performers (Exam > 80)":
            filtered_data = self.data[self.data['exam_score'] > 80]
        elif filter_option == "Study Hours > 4":
            filtered_data = self.data[self.data['study_hours_per_day'] > 4]
        elif filter_option == "Sleep Hours > 7":
            filtered_data = self.data[self.data['sleep_hours'] > 7]
        else:
            filtered_data = self.data
        # Create a dialog to display the table
        dialog = QDialog(self)
        dialog.setWindowTitle(f"Student Data - {filter_option}")
        dialog.setMinimumSize(800, 600)
        dialog.setStyleSheet("""
            QDialog {
                background-color: #1e1e1e;
            }
            QTableWidget {
                background-color: #242424;
                color: white;
                gridline-color: #3d5afe;
                border: 1px solid #3d5afe;
                border-radius: 10px;
                selection-background-color: #3d5afe;
                selection-color: white;
            }
            QTableWidget::item {
                padding: 5px;
                border-bottom: 1px solid rgba(61, 90, 254, 0.3);
            }
            QTableWidget::item:selected {
                background-color: #3d5afe;
            }
            QHeaderView::section {
                background-color: #121212;
                color: white;
                padding: 6px;
                border: 1px solid #3d5afe;
                font-weight: bold;
            }
            QPushButton {
                background: qlineargradient(x1:0, y1:0, x2:0, y2:1, stop:0
#3d5afe, stop:1 #1a237e);
                color: white;
                border: none;
                border-radius: 8px;
                padding: 10px 20px;
              font-weight: bold;
                min-height: 35px;
            }
            QPushButton:hover {
                background: qlineargradient(x1:0, y1:0, x2:0, y2:1, stop:0
#536dfe, stop:1 #283593);
            }
            QLabel {
                color: white;
                font-size: 12px;
            }
        """)
        # Create a layout for the dialog
        layout = QVBoxLayout(dialog)
        layout.setContentsMargins(20, 20, 20, 20)
    layout.setSpacing(15)
    # Add a header with stats about the filtered data
    header_layout = QHBoxLayout()
    record_count = QLabel(f"Records: {len(filtered_data)}")
    avg_study = QLabel(f"Avg Study: 
{filtered_data['study_hours_per_day'].mean():.2f} hrs")
    avg_score = QLabel(f"Avg Score: 
{filtered_data['exam_score'].mean():.2f}")
     header_laysout.addWidget(record_count)
    header_layout.addWidget(avg_study)
    header_layout.addWidget(avg_score)
    header_layout.addStretch()
    layout.addLayout(header_layout)
    # Create a table widget to display the data
    table = QTableWidget()
    # Set the number of rows and columns
    rows, cols = filtered_data.shape
    table.setRowCount(rows)
    table.setColumnCount(cols)
    # Set the column headers
    table.setHorizontalHeaderLabels(filtered_data.columns)
    # Fill the table with data
    for i in range(rows):
        for j in range(cols):
            value = str(filtered_data.iloc[i, j])
            item = QTableWidgetItem(value)
            # Center align the text
            item.setTextAlignment(Qt.AlignCenter)
            # Color high exam scores in green
            if j == 15:  # exam_score column
                try:
                    score = float(value)
                    if score >= 90:
                        item.setForeground(QColor(0, 255, 0))  # Green for high 
scores
                    elif score >= 70:
                        item.setForeground(QColor(255, 165, 0))  # Orange for 
medium scores
                    elif score < 50:
                        item.setForeground(QColor(255, 0, 0))  # Red for low 
scores
                except:
                    pass
            table.setItem(i, j, item)
    # Resize columns to contents
    header = table.horizontalHeader()
    for i in range(cols):
        header.setSectionResizeMode(i, 
QHeaderView.ResizeToContents)
    # Make the table scrollable and stretch to fill available space
    table.setSizePolicy(QSizePolicy.Expanding, 
QSizePolicy.Expanding)
    layout.addWidget(table)
    # Add a close button
    close_button = QPushButton("Close")
    close_button.clicked.connect(dialog.close)
    layout.addWidget(close_button)
    # Show the dialog
    dialog.exec_()
    # Add summary to output text
    summary = (
        f"Data Summary for {filter_option}:\n\n"
```

6.Conclusion

The analysis conducted through the *Student Habits and Performance Analysis* application reveals significant insights into how various habits affect academic performance. Study hours and sleep patterns showed a strong positive correlation with exam scores and attendance, indicating that consistent study and adequate rest contribute to better academic outcomes. Categorical analysis highlighted notable trends in behavior across gender and lifestyle factors, while regression models and confidence intervals validated the reliability of predictions. Through intuitive visualizations and comprehensive statistical tools, the application effectively demonstrates that student success is closely linked to balanced daily routines and disciplined study habits.