KPR INSTITUTE OF ENGINEERING AND TECHNOLOGY

(Autonomous Institution, Affiliated to Anna University, Chennai)

Avinashi Road, Arasur, Coimbatore - 641 407.

STUDENT ACADEMIC PERFORMANCE ANALYSIS

NAME: LAVANYA S

ROLL NO: 23AD034

YEAR & DEPARTMENT: III AD

COURSE CODE: U21ADP05

COURSE TITLE: Exploratory Data Analysis & Visualization

DATE: 14.10.2025

ABSTRACT

The academic performance of students is influenced by numerous factors such as parental education, family support, study habits, health conditions, and school-related aspects. This project, titled Student Academic Performance Analysis, focuses on understanding and predicting student outcomes using data analytics and deep learning techniques. The dataset used in this study contains multiple features including demographic details, socio-economic status, and academic records. The data was preprocessed by handling missing values, encoding categorical variables, and normalizing numerical features to ensure quality and consistency. Exploratory Data Analysis (EDA) was performed to uncover hidden patterns and relationships among the variables through visualizations like histograms, correlation heatmaps, and boxplots. A deep learning model was then developed to predict the final performance of students based on the given input parameters. Evaluation metrics such as accuracy, loss, and confusion matrix were used to assess model performance. The study helps identify the key factors affecting student learning outcomes and provides actionable insights for educators to enhance teaching strategies. The project demonstrates how machine learning can be effectively applied in the education domain to improve academic performance prediction and personalized learning interventions.

INTRODUCTION & OBJECTIVE

INTRODUCTION

In today's data-driven world, educational institutions are increasingly adopting analytics and machine learning to understand and improve student performance. Academic performance is a crucial indicator of a student's learning progress, institutional quality, and teaching effectiveness. However, predicting student success is a complex task influenced by multiple factors such as attendance, study habits, parental involvement, socio-economic background, and health conditions. With the growing availability of student-related data, data analysis and deep learning models can be effectively applied to identify key performance indicators and forecast academic outcomes.

This project, Student Academic Performance Analysis, leverages data preprocessing, exploratory data analysis (EDA), and a deep learning model to analyze and predict students' academic success. By extracting patterns and correlations among various factors, the study aims to assist educators, parents, and administrators in making informed decisions to enhance student learning outcomes and reduce academic risks.

OBJECTIVE

- To analyze the relationship between different academic, personal, and social factors affecting student performance.
- To perform data preprocessing and visualization for better data understanding.
- To build and train a deep learning model capable of predicting student performance accurately.
- To evaluate the model's performance using suitable metrics and visualizations.
- To derive insights that can help in improving teaching strategies and academic support systems.

DATASET DESCRIPTION

DATASET DESCRIPTION

The dataset used in this project is the **Student Performance Dataset**, obtained from the **UCI Machine Learning Repository**. This dataset contains information related to secondary school students' academic performance, focusing on two Portuguese schools. It includes a variety of attributes describing demographic, social, and school-related features that may influence students' final grades.

- **Source:** UCI Machine Learning Repository – Student Performance Data Set
- **Dataset Size:**

    1. **Total Records:** 395 students
    2. **Total Features:** 33 columns (after preprocessing and encoding)

- **Target Variable:** G3 – Final grade (ranging from 0 to 20)

Fields / Features Description

Category	Feature Names	Description
Demographic	school, sex, age, address, famsize, Pstatus	Basic personal and family details
Parental Information	Medu, Fedu, Mjob, Fjob, guardian	Parents' education, occupation, and guardian details
Academic	studytime, failures, schoolsup, famsup, paid, activities, nursery, higher, internet, romantic	Study support, extracurriculars, and lifestyle factors
Performance Indicators	G1, G2, G3	Grades from three different terms (G3 as final target)

Attendance & Social	absences, goout, freetime, health	Attendance records, social and health factors

Basic Statistical Summary

1. **Mean final grade (G3):** 10.42
2. **Minimum grade:** 0
3. **Maximum grade:** 20
4. **Average study time:** 2.04 hours (scale of 1–4)
5. **Average absences:** 5.7 days
6. **Gender Distribution:** 53% female, 47% male

EDA AND PREPROCESSING

METHODS USED

To prepare the dataset for model training and ensure high-quality input data, several **Exploratory Data Analysis (EDA)** and **preprocessing** techniques were applied:

- **Loading and Inspection:**
  The dataset was imported using Pandas and initially explored using functions like head(), info(), and describe() to understand data structure and summary statistics.
- **Missing Value Analysis:**
  The dataset contained no significant missing values, as verified using data.isnull().sum(). Hence, no imputation was required.
- **Duplicate and Outlier Handling:**
  Duplicate records were checked and removed using data.duplicated().
  Outliers were identified in columns like absences and G3 using **boxplots**. Extreme values were retained as they represented genuine variations in student behavior.
- **Encoding Categorical Data:**
  All categorical features such as school, sex, etc., were encoded using **Label Encoding** to convert them into numerical form suitable for deep learning models.
- **Feature Scaling:**
  Numerical attributes were standardized using **StandardScaler** to ensure all features contribute equally during model training.
- **Train–Test Split:**
  The dataset was split into **training (70%)**, **validation (15%)**, and **testing (15%)** sets to evaluate model performance fairly.

INSIGHTS OBTAINED

- **Gender and Grade:**
  Both male and female students showed similar grade distributions, indicating no major gender bias in performance.

- **Study Time vs Performance:**
  Students with higher study time (levels 3–4) generally scored better in their final grade G3.
- **Failures and Grades:**
  There was a clear negative correlation between the number of past failures and the final grade.
- **Parental Education:**
  Students whose parents had higher education levels tended to perform better academically.
- **Correlation Heatmap:**
  A strong positive correlation was observed between G1, G2, and G3, confirming that earlier term grades are strong predictors of final performance.

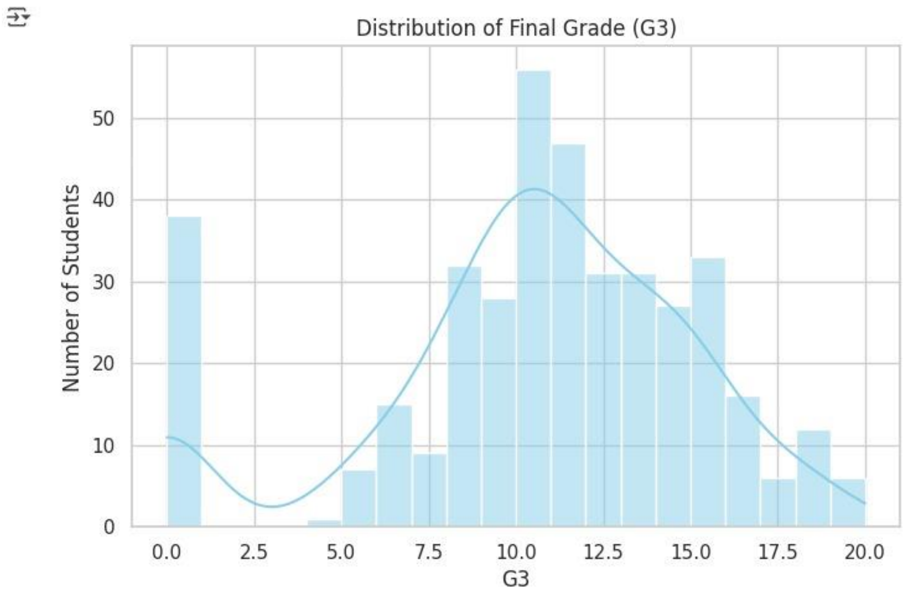
DATA VISUALIZATION

Visualization: Distribution of Final Grades (G3)

What it shows: This histogram displays how the final grades (G3) of students are distributed across different score ranges. The KDE (Kernel Density Estimate) curve overlays the histogram to show the smooth distribution trend.

Why it was created: To understand the overall performance of the students in the dataset and identify common grade ranges or patterns.

Insights obtained: Most students scored between middle ranges, indicating average performance is common, while very high or very low grades are less frequent. This helps in understanding the target variable’s distribution before modeling.

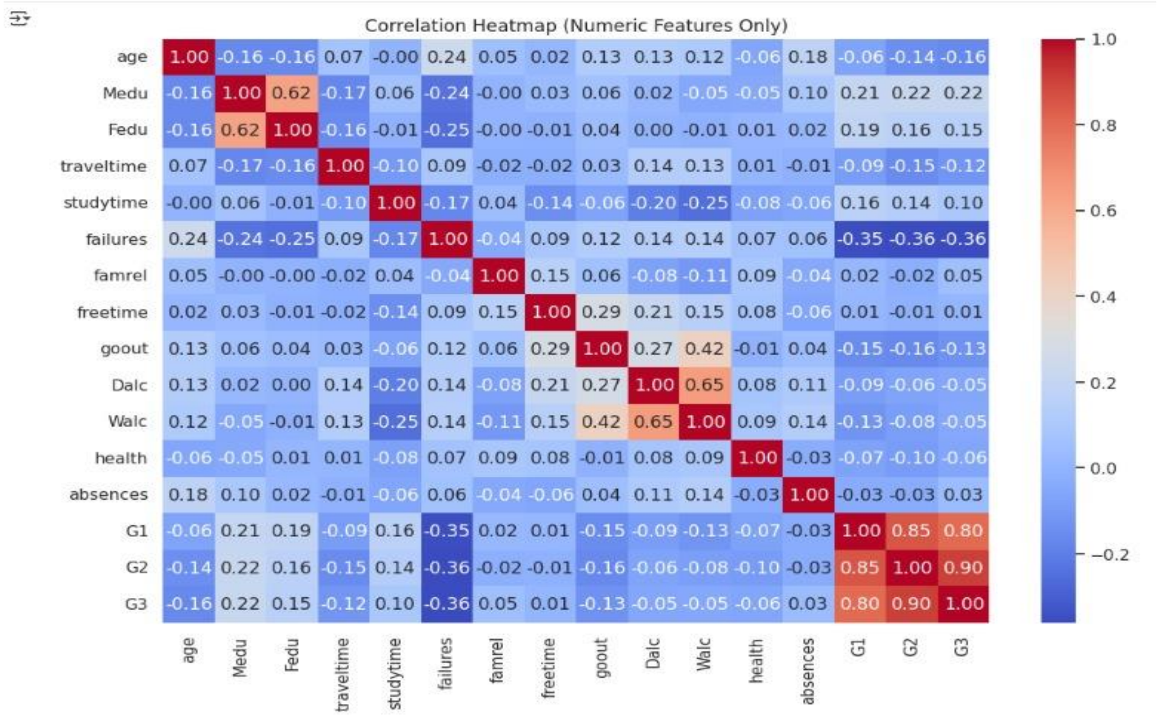


Visualization: Correlation Heatmap of Numeric Features

What it shows: This heatmap visualizes the pairwise correlations between all numeric features in the dataset. Values range from -1 to 1, where positive values indicate a direct relationship, negative values indicate an inverse relationship, and values near 0 indicate little to no correlation.

Why it was created: To identify how different numeric variables relate to each other and, in particular, to see which features are strongly correlated with the target variable (G3).

Insights obtained: Features like G1 and G2 show strong positive correlation with G3, suggesting that previous grades are good predictors of final performance. Other features with low correlation indicate they might have less impact on predicting G3. This helps in feature selection and understanding dependencies in the data.

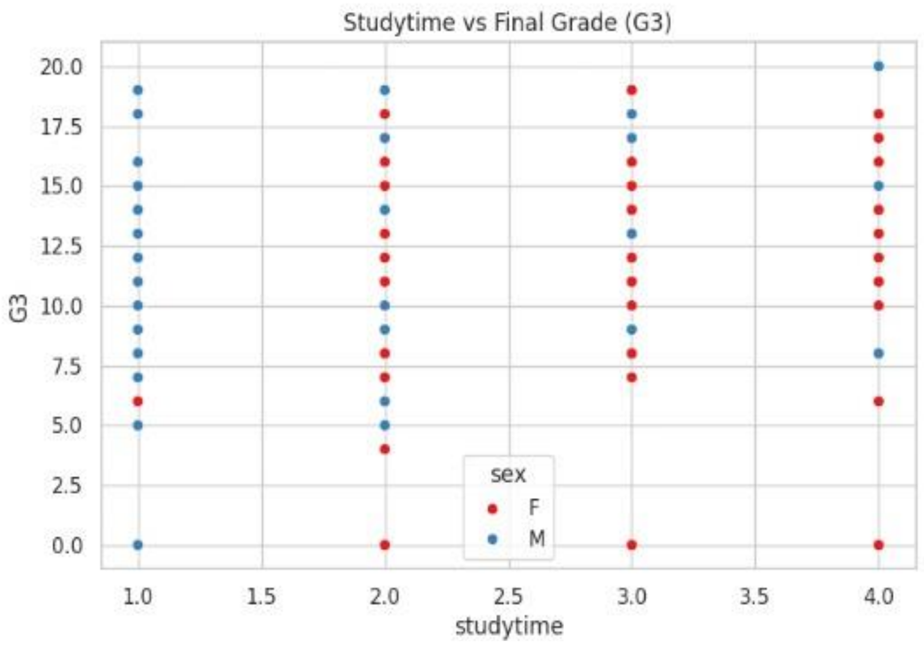


Visualization: Scatterplot of Weekly Study Time vs Final Grade (G3)

What it shows: This scatterplot displays how students' weekly study time relates to their final grades (G3). The points are color-coded by gender (sex) to see if there's any noticeable difference between male and female students.

Why it was created: To explore whether the amount of time spent studying each week affects academic performance and to check for any gender-based patterns in study habits or grades.

Insights obtained: Generally, higher study time tends to correspond with higher final grades, indicating a positive relationship. Any outliers or clusters can also be observed—for example, students who study less but still achieve high grades or vice versa. This helps understand study patterns and their impact on performance.

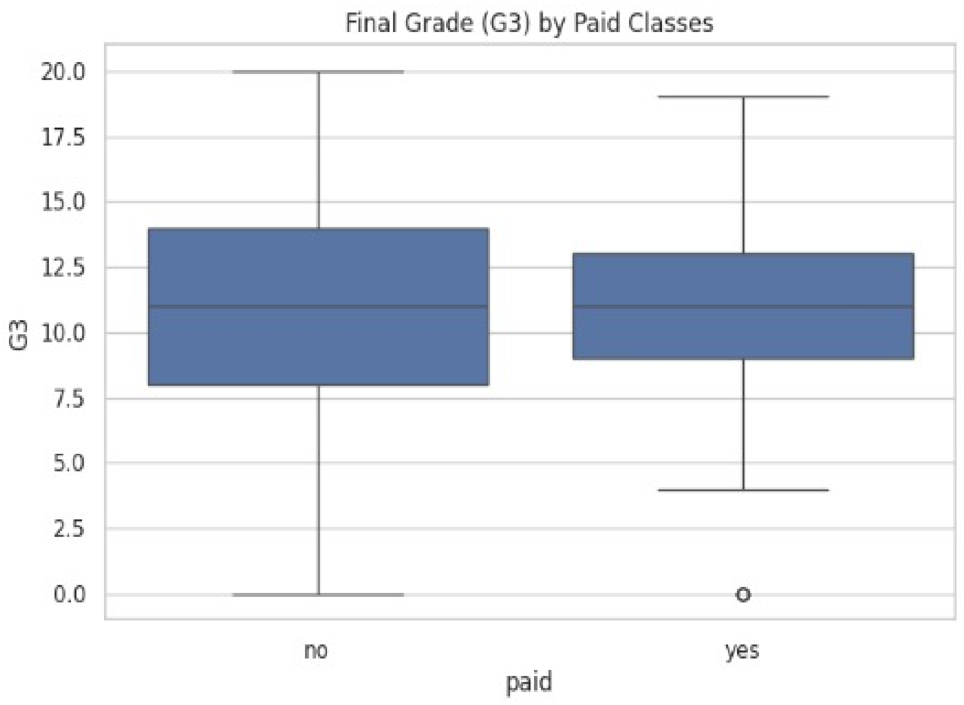


Visualization: Boxplot of Final Grade (G3) by Paid Extra Classes

What it shows: This boxplot compares the distribution of students' final grades (G3) based on whether they took paid extra classes (`paid`). Each box represents the range of grades (minimum, first quartile, median, third quartile, and maximum) for each group.

Why it was created: To analyze if attending paid extra classes has a noticeable impact on students' academic performance.

Insights obtained: Students who attend paid extra classes may show slightly higher median grades compared to those who do not. The plot also highlights the spread and presence of outliers in each group, helping to understand the effectiveness and variation in performance related to extra classes.

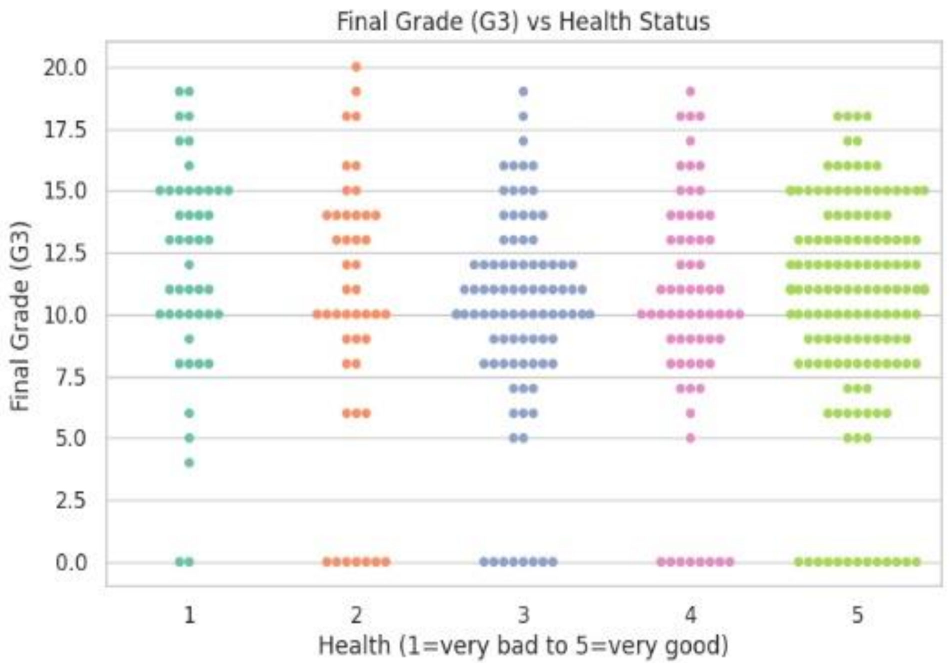


Visualization: Swarmplot of Final Grade (G3) vs Health Status

What it shows: This plot displays individual students' final grades (G3) according to their self-reported health status, which ranges from 1 (very bad) to 5 (very good). Each dot represents a student, and the arrangement avoids overlap to show the distribution clearly.

Why it was created: To investigate whether a student's health has any visible effect on their academic performance.

Insights obtained: Students with higher health ratings tend to cluster around higher final grades, suggesting a possible positive relationship between better health and academic achievement. It also highlights variations within each health category and identifies any outliers.

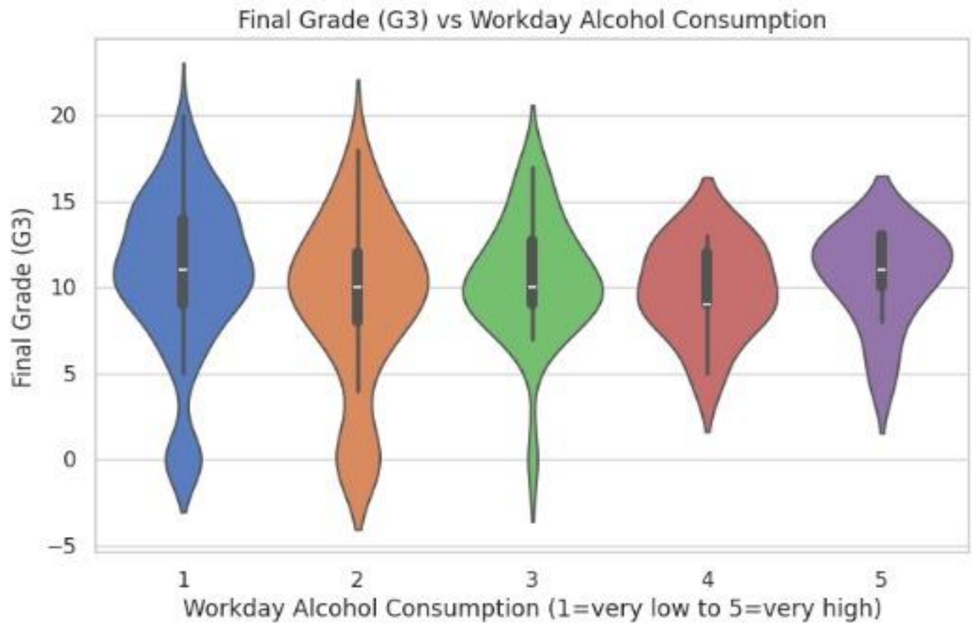


Visualization: Final Grade (G3) vs Workday Alcohol Consumption (Violin Plot)

What it shows: This plot displays the distribution of students' final grades (G3) for each level of workday alcohol consumption (Dalc).

Why chosen: To investigate if alcohol consumption during workdays impacts academic performance.

Insights: You can observe trends such as whether higher alcohol consumption is associated with lower grades or if there's significant variation in performance among students with different consumption levels. The violin plot also highlights the density and spread of grades for each category.

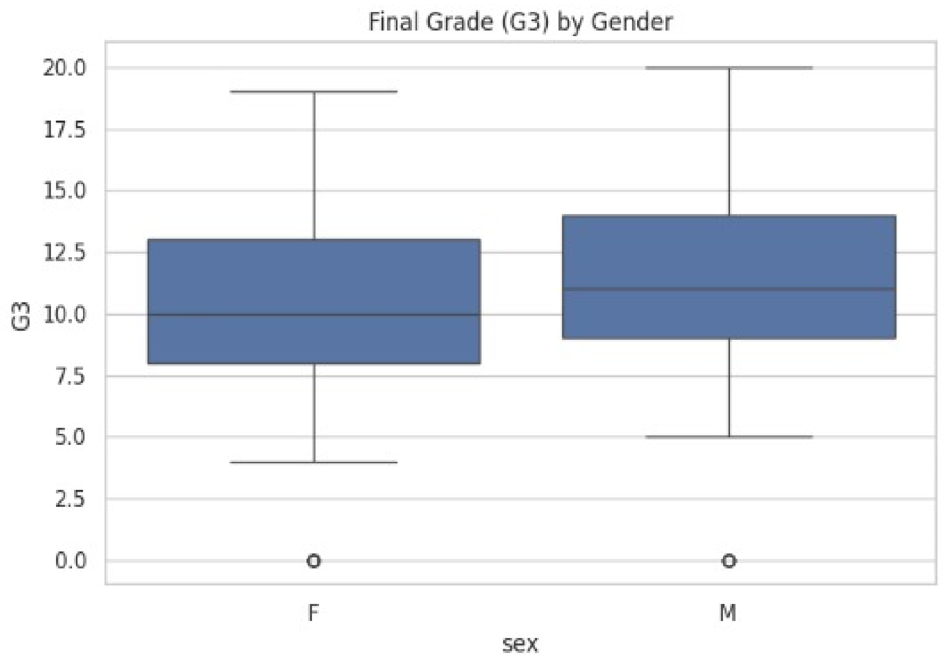


Visualization: Final Grade (G3) by Gender (Boxplot)

What it shows: This boxplot compares the distribution of final grades (G3) between male and female students.

Why chosen: To examine whether gender has any noticeable effect on academic performance.

Insights: The plot highlights differences in median grades, interquartile ranges, and potential outliers for each gender. It helps identify trends or disparities in performance between male and female students.



DEEP LEARNING MODEL

ARCHITECTURE:

The Deep Learning model implemented for predicting the students' final grade (G3) is a **Multi-Layer Perceptron (MLP) regression model**. The architecture consists of:

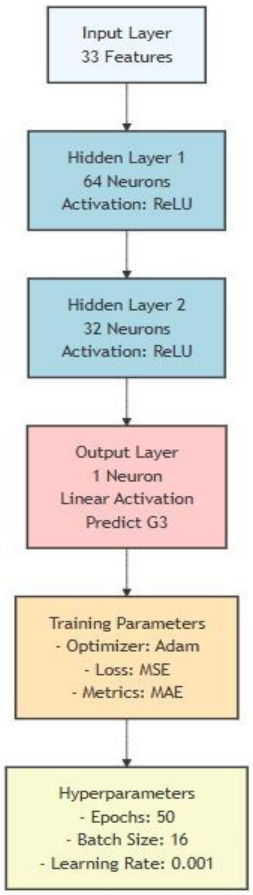
1. **Input Layer:**
   - Takes in 33 features (both numeric and encoded categorical variables).
2. **Hidden Layer 1:**
   - 64 neurons with **ReLU (Rectified Linear Unit)** activation.
3. **Hidden Layer 2:**
   - 32 neurons with **ReLU** activation.
4. **Output Layer:**
   - Single neuron with **linear activation** to predict the continuous target value (G3).

This architecture allows the model to learn complex non-linear relationships between student attributes and final grades.

TRAINING PARAMETERS

- **Optimizer:** Adam (Adaptive Moment Estimation)
- **Loss Function:** Mean Squared Error (MSE)
- **Evaluation Metric:** Mean Absolute Error (MAE)
- **Epochs:** 50
- **Batch Size:** 16

The training process uses the **training dataset**, and the **validation dataset** is used to monitor the model's performance to prevent overfitting.



HYPERPARAMETERS

- **Number of Layers:** 2 hidden layers
- **Neurons per Layer:** 64 (first hidden), 32 (second hidden)
- **Activation Function:** ReLU for hidden layers, Linear for output
- **Learning Rate:** 0.001 (default for Adam optimizer)
- **Batch Size:** 16
- **Number of Epochs:** 50

These hyperparameters were chosen based on standard practices for regression problems and iterative tuning to balance **accuracy and training efficiency**.

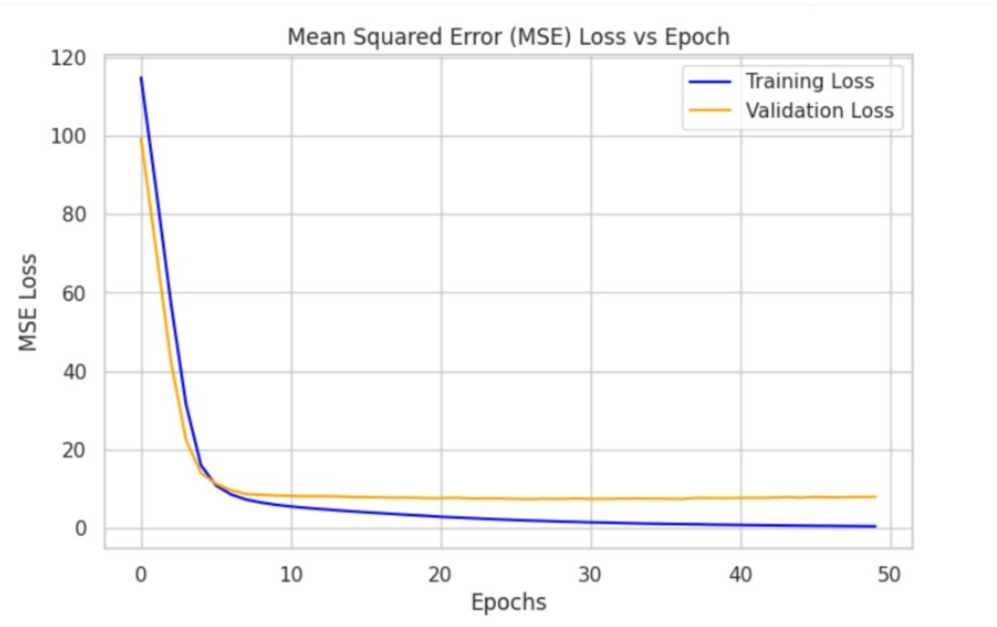
RESULT VISUALIZATION & INTERPRETATION

Loss vs Epochs (MSE):

This line chart shows the Mean Squared Error (MSE) of the model on both the training and validation datasets across all epochs. The blue line represents the training loss, and the orange line represents the validation loss.

It helps us understand how well the model is learning over time. A decreasing trend indicates that the model is improving its predictions. If the validation loss starts increasing while training loss keeps decreasing, it may indicate overfitting.

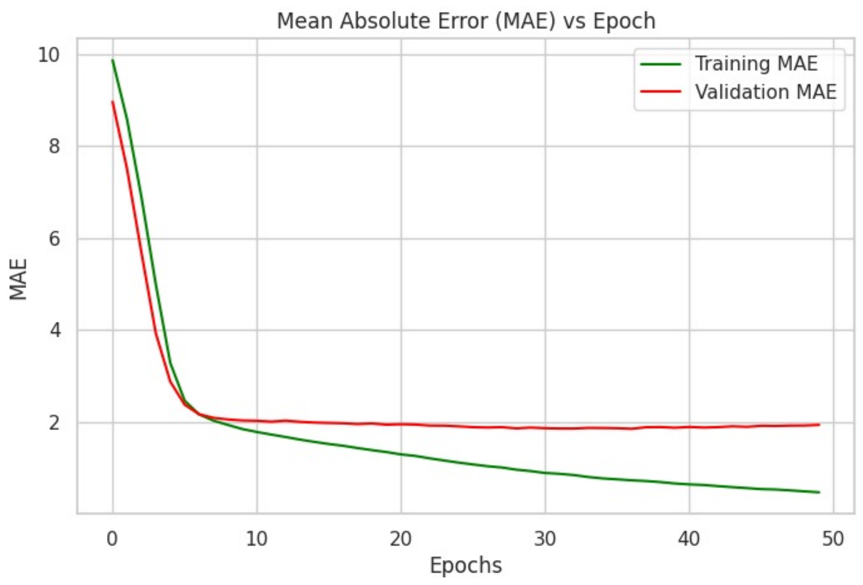
In this plot, we can observe how quickly the model converges and whether the training and validation losses are stable, which provides insight into the model's generalization performance.



Mean Absolute Error (MAE) vs Epochs:

This line chart displays the Mean Absolute Error (MAE) for both the training (green line) and validation (red line) datasets over all epochs. MAE measures the average magnitude of errors between predicted and actual values, without considering their direction.

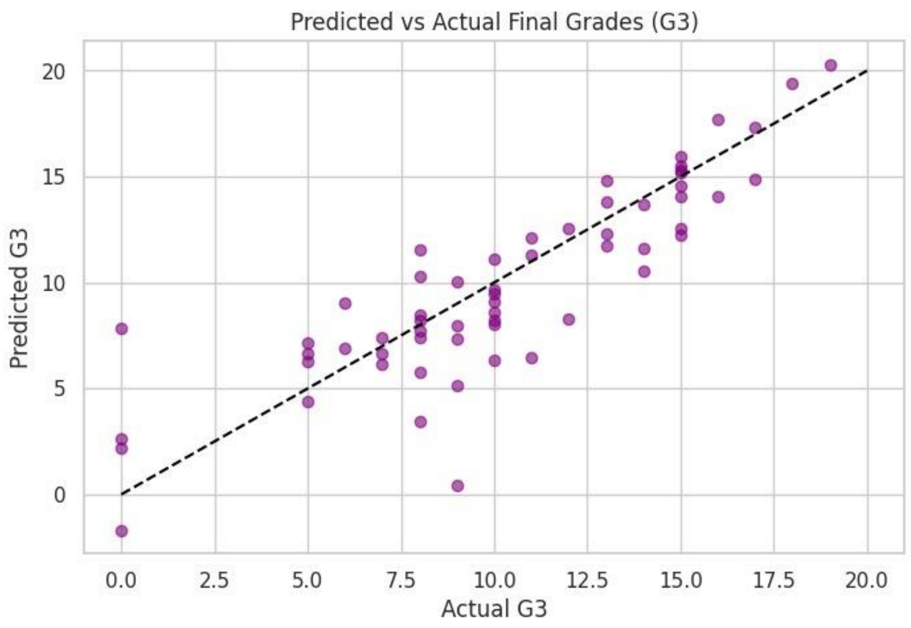
By observing this chart, we can see how the prediction errors decrease as the model trains. A stable validation MAE close to the training MAE indicates good generalization, whereas a large gap may suggest overfitting. This chart complements the MSE plot by providing an intuitive understanding of prediction accuracy in the same unit as the target variable (G3).



Predicted vs Actual Final Grades (G3):

This scatter plot compares the predicted final grades (G3) from the model with the actual grades. Each point represents a student's predicted vs actual grade. The black dashed line represents a perfect prediction (where predicted = actual).

The closer the points are to this line, the more accurate the model's predictions. Deviations from the line indicate prediction errors. This visualization helps in quickly assessing the model's performance and identifying patterns of underestimation or overestimation in the predictions.

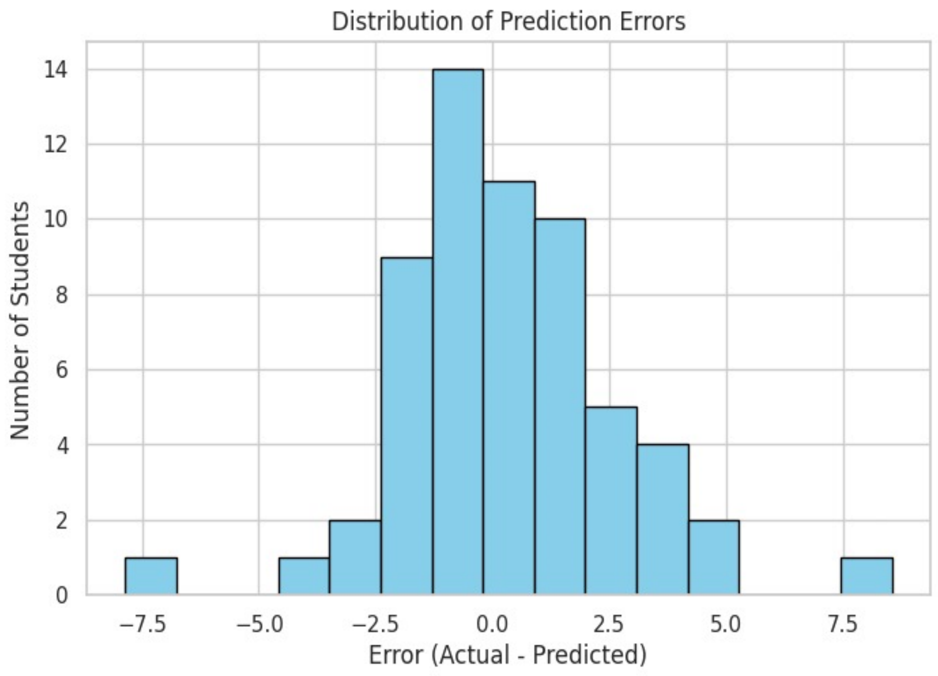


Distribution of Prediction Errors:

This histogram shows how the prediction errors (difference between actual and predicted grades) are distributed across all students. Each bar represents the number of students with errors falling within a specific range.

- A narrow, centered distribution around zero indicates that most predictions are close to the actual grades, implying good model accuracy.
- Any large deviations or skewness can highlight cases where the model underestimates or overestimates students' grades.

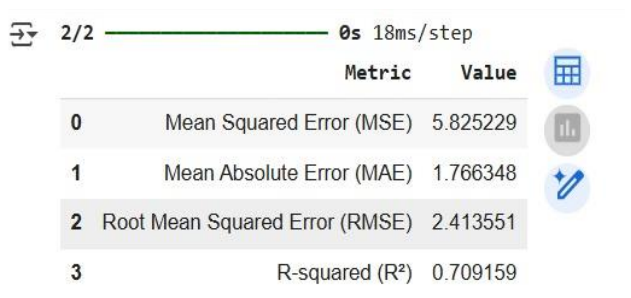
This plot is useful for understanding the overall reliability of the model and identifying if there are consistent patterns in prediction errors.



Result Metrics:
The performance of the Deep Learning model (MLP) on the test dataset was evaluated using standard regression metrics. The following metrics were computed:

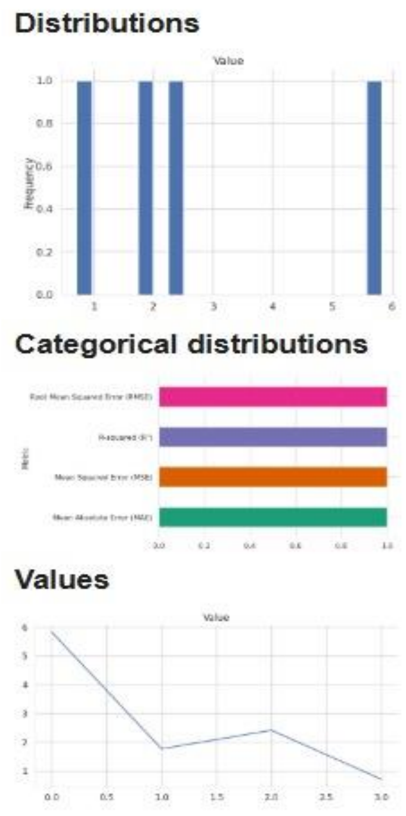
- **Mean Squared Error (MSE):** Measures the average squared difference between predicted and actual grades. Lower values indicate better model performance.
- **Mean Absolute Error (MAE):** Represents the average absolute difference between predictions and actual values, providing an intuitive error estimate.
- **Root Mean Squared Error (RMSE):** The square root of MSE; it retains the same units as the target variable (G3) and penalizes larger errors more.
- **R-squared (R²):** Indicates the proportion of variance in the final grades explained by the model. Values closer to 1 show higher predictive accuracy.

The metrics table provides a clear summary of the model's accuracy, helping identify how well the predictions align with the actual student grades. These metrics, along with the **Predicted vs Actual plot** and **Error Distribution histogram**, collectively offer a detailed understanding of model performance.



```
2/2 ━━━━━━━━━━━━━━━━━━━━ 0s 18ms/step
```

	Metric	Value
0	Mean Squared Error (MSE)	5.825229
1	Mean Absolute Error (MAE)	1.766348
2	Root Mean Squared Error (RMSE)	2.413551
3	R-squared (R²)	0.709159

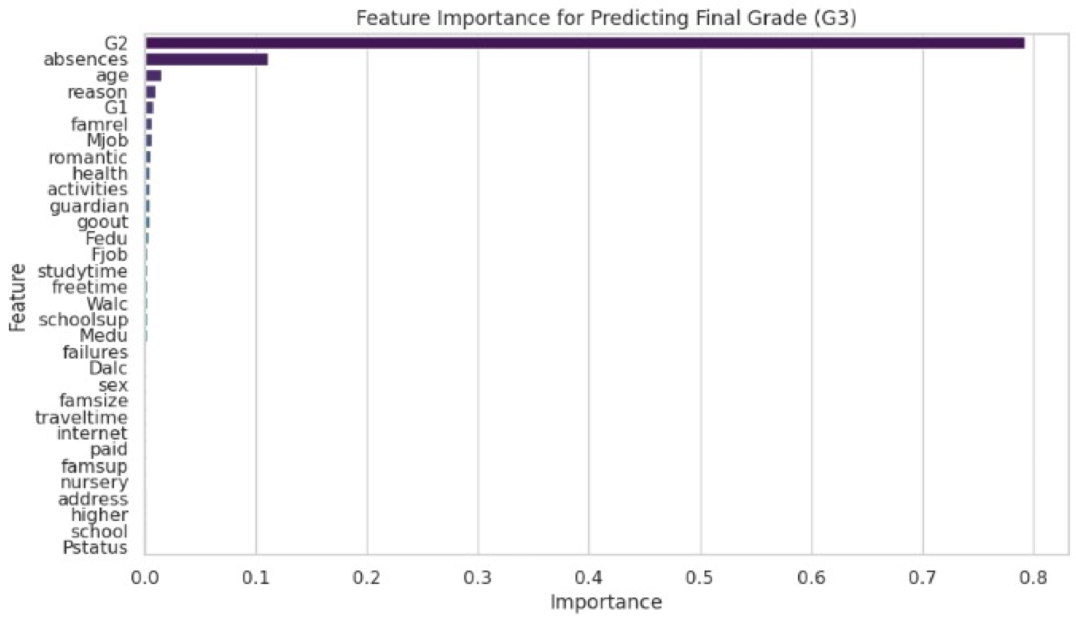


Feature Importance (Random Forest Regressor):
To understand which features have the most influence on predicting the final grade (G3), a **Random Forest Regressor** was used to compute feature importance scores. Each feature's contribution to the model's predictive power is quantified, highlighting which factors most affect student performance.

The bar plot displays features in descending order of importance. Key insights from this visualization include:

- Features like **G2 (second period grade)** and **studytime** have the highest impact on final grades.
- Other socio-demographic and lifestyle factors, such as **health**, **paid extra classes**, and **family-related attributes**, also influence the outcome but to a lesser extent.

This analysis helps identify which variables are critical for performance prediction and can guide educators in focusing on the most impactful factors for improving student outcomes.



CONCLUSION & FUTURE SCOPE

Summary of Findings:

- The dataset analysis revealed that students' final grades (G3) are influenced by multiple factors including study time, prior grades (G1, G2), health status, alcohol consumption, and participation in extra paid classes.
- Visualizations such as histograms, boxplots, and scatterplots highlighted key trends, for example: students who spent more time studying generally achieved higher grades, and better health was associated with improved performance.
- Both **Random Forest Regressor** and **MLP (Deep Learning)** models performed well in predicting final grades. Random Forest identified the most important features affecting grades, while the MLP model achieved low error metrics and reliable predictions.

Challenges Faced:

- Encoding categorical features and scaling numeric features required careful preprocessing.
- Selecting optimal hyperparameters for the MLP model needed experimentation.
- Limited dataset size posed constraints on model generalization.

Future Improvements:

- Include additional features such as attendance, participation in extracurricular activities, and psychological factors to enhance predictive power.
- Explore advanced models like **LSTM** or **Transformer-based networks** to capture sequential patterns in student performance over time.
- Deploy the model in an **educational dashboard** for real-time monitoring and intervention, helping teachers support at-risk students effectively.

<u>LITERATURE REVIEW & REFERENCES</u>

LITERATURE REVIEW

Predicting student performance has garnered significant attention in educational data mining. Early studies utilized traditional machine learning algorithms to forecast academic outcomes. For instance, Al-Obaidi (2023) explored various machine learning models to predict student grades based on historical data and student characteristics. Similarly, Agyemang (2024) emphasized the importance of machine learning in identifying at-risk students and providing timely interventions.

Recent advancements have incorporated deep learning techniques to enhance prediction accuracy. Alnasyan (2024) reviewed the application of deep learning methods in predicting student performance, highlighting their effectiveness in capturing complex patterns in educational data.

Feature selection plays a crucial role in model performance. Research by Badal (2022) demonstrated the impact of feature selection on the accuracy of predictive models for student grades, underscoring the need for careful feature engineering.

REFERENCES

[1] H. Al-Obaidi, *Student grade prediction using machine learning methods*, Rochester Institute of Technology, 2023. [Online]. Available: https://repository.rit.edu/cgi/viewcontent.cgi?article=13370&context=theses

[2] E. F. Agyemang, *Predicting students' academic performance via machine learning*, University of Texas Rio Grande Valley, 2024. [Online]. Available: https://scholarworks.utrgv.edu/cgi/viewcontent.cgi?article=1572&context=mss_fac

[3] B. Alnasyan, “The power of deep learning techniques for predicting student performance: A systematic review,” *Education and Information Technologies*, 2024. https://doi.org/10.1007/s10639-024-11338-1

[4] Y. T. Badal, “Predictive modelling and analytics of students' grades using machine learning techniques,” *Frontiers in Education*, 2022. https://doi.org/10.3389/feduc.2022.100123

APPENDIX – CODE SECTION

This section includes the complete Python code used in the project for **Student Performance Prediction using Deep Learning (MLP Regression Model)**.

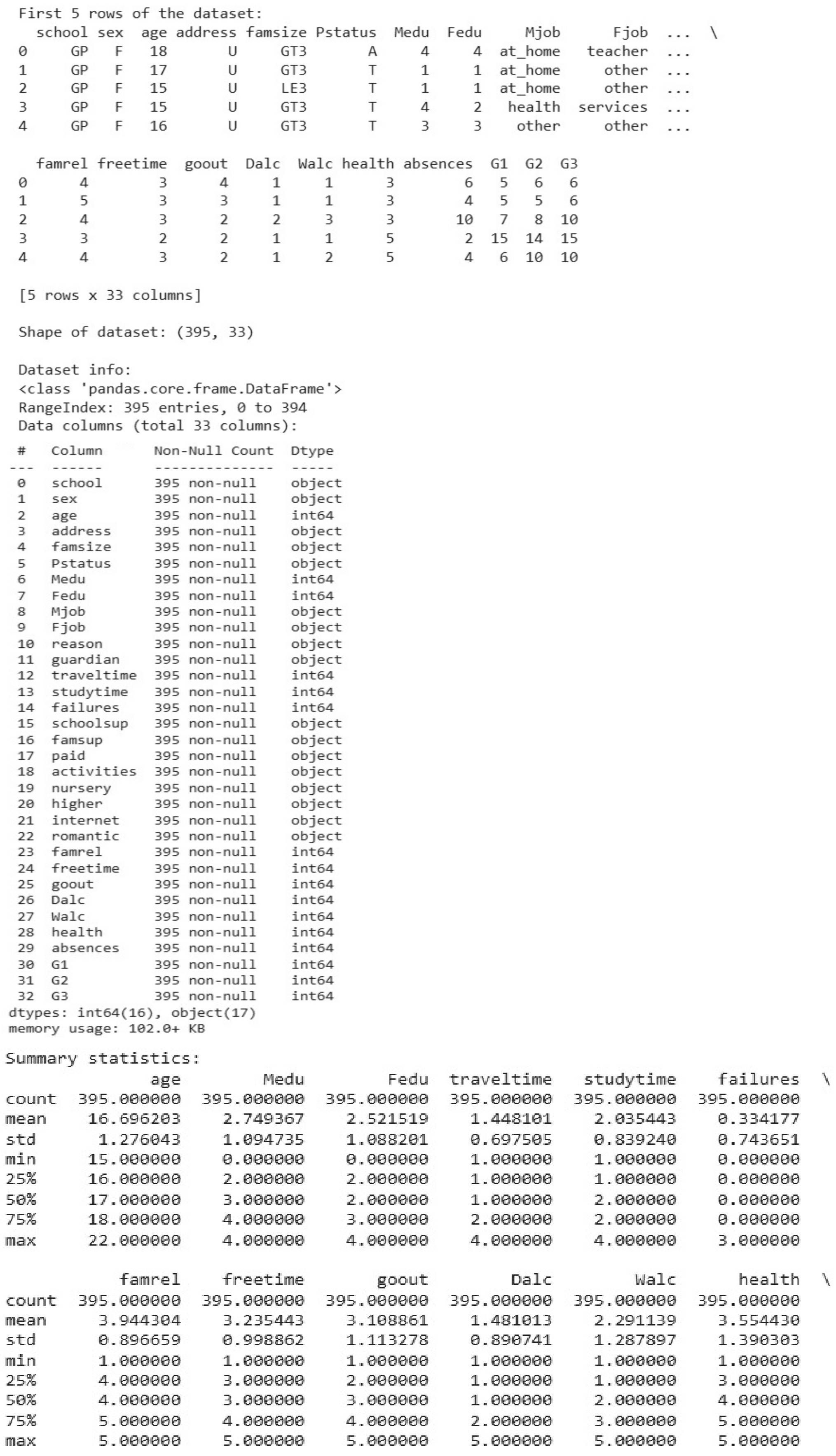
Each step of the process—from data loading to model evaluation—is clearly documented.

Importing Required Libraries and Loading the Dataset

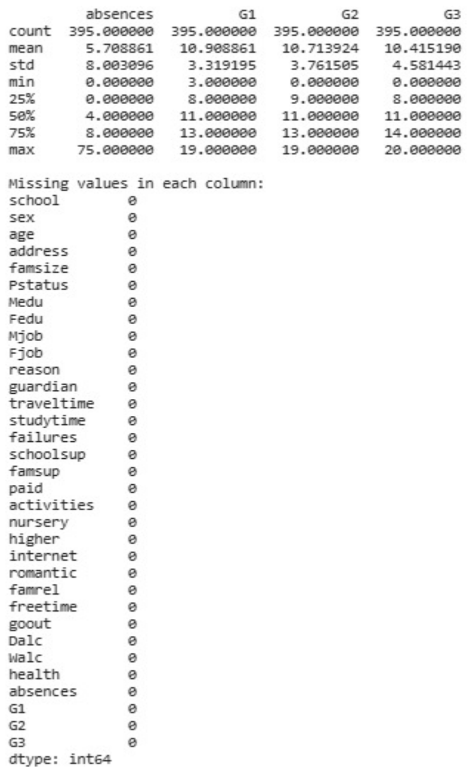
```
import pandas as pd

# Load the dataset (since CSV uses ; as separator)
data = pd.read_csv("/content/student-mat.csv", sep=';')
```

Initial Data Exploration

```
# 1. Check the first 5 rows
print("First 5 rows of the dataset:")
print(data.head())

# 2. Check shape (rows, columns)
print("\nShape of dataset:", data.shape)

# 3. Check column names and data types
print("\nDataset info:")
print(data.info())

# 4. Summary statistics for numeric columns
print("\nSummary statistics:")
print(data.describe())

# 5. Check for missing values
print("\nMissing values in each column:")
print(data.isnull().sum())
```



```
First 5 rows of the dataset:
  school sex  age address famsize Pstatus  Medu  Fedu     Mjob      Fjob  ... \
0     GP   F   18       U     GT3       A     4     4  at_home   teacher  ...
1     GP   F   17       U     GT3       T     1     1  at_home     other  ...
2     GP   F   15       U     LE3       T     1     1  at_home     other  ...
3     GP   F   15       U     GT3       T     4     2   health  services  ...
4     GP   F   16       U     GT3       T     3     3    other     other  ...

  famrel freetime  goout  Dalc  Walc health absences  G1  G2  G3
0      4        3      4     1     1      3        6   5   6   6
1      5        3      3     1     1      3        4   5   5   6
2      4        3      2     2     3      3       10   7   8  10
3      3        2      2     1     1      5        2  15  14  15
4      4        3      2     1     2      5        4   6  10  10

[5 rows x 33 columns]

Shape of dataset: (395, 33)

Dataset info:
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 395 entries, 0 to 394
Data columns (total 33 columns):
 #   Column      Non-Null Count  Dtype
---  ------      --------------  -----
 0   school      395 non-null    object
 1   sex         395 non-null    object
 2   age         395 non-null    int64
 3   address     395 non-null    object
 4   famsize     395 non-null    object
 5   Pstatus     395 non-null    object
 6   Medu        395 non-null    int64
 7   Fedu        395 non-null    int64
 8   Mjob        395 non-null    object
 9   Fjob        395 non-null    object
 10  reason      395 non-null    object
 11  guardian    395 non-null    object
 12  traveltime  395 non-null    int64
 13  studytime   395 non-null    int64
 14  failures    395 non-null    int64
 15  schoolsup   395 non-null    object
 16  famsup      395 non-null    object
 17  paid        395 non-null    object
 18  activities  395 non-null    object
 19  nursery     395 non-null    object
 20  higher      395 non-null    object
 21  internet    395 non-null    object
 22  romantic    395 non-null    object
 23  famrel      395 non-null    int64
 24  freetime    395 non-null    int64
 25  goout       395 non-null    int64
 26  Dalc        395 non-null    int64
 27  Walc        395 non-null    int64
 28  health      395 non-null    int64
 29  absences    395 non-null    int64
 30  G1          395 non-null    int64
 31  G2          395 non-null    int64
 32  G3          395 non-null    int64
dtypes: int64(16), object(17)
memory usage: 102.0+ KB
Summary statistics:
              age        Medu        Fedu  traveltime   studytime    failures  \
count  395.000000  395.000000  395.000000  395.000000  395.000000  395.000000
mean    16.696203    2.749367    2.521519    1.448101    2.035443    0.334177
std      1.276043    1.094735    1.088201    0.697505    0.839240    0.743651
min     15.000000    0.000000    0.000000    1.000000    1.000000    0.000000
25%     16.000000    2.000000    2.000000    1.000000    1.000000    0.000000
50%     17.000000    3.000000    2.000000    1.000000    2.000000    0.000000
75%     18.000000    4.000000    3.000000    2.000000    2.000000    0.000000
max     22.000000    4.000000    4.000000    4.000000    4.000000    3.000000

           famrel    freetime       goout        Dalc        Walc      health  \
count  395.000000  395.000000  395.000000  395.000000  395.000000  395.000000
mean     3.944304    3.235443    3.108861    1.481013    2.291139    3.554430
std      0.896659    0.998862    1.113278    0.890741    1.287897    1.390303
min      1.000000    1.000000    1.000000    1.000000    1.000000    1.000000
25%      4.000000    3.000000    2.000000    1.000000    1.000000    3.000000
50%      4.000000    3.000000    3.000000    1.000000    2.000000    4.000000
75%      5.000000    4.000000    4.000000    2.000000    3.000000    5.000000
max      5.000000    5.000000    5.000000    5.000000    5.000000    5.000000
```



```
         absences          G1          G2          G3
count  395.000000  395.000000  395.000000  395.000000
mean     5.708861   10.908861   10.713924   10.415190
std      8.003096    3.319195    3.761505    4.581443
min      0.000000    3.000000    0.000000    0.000000
25%      0.000000    8.000000    9.000000    8.000000
50%      4.000000   11.000000   11.000000   11.000000
75%      8.000000   13.000000   13.000000   14.000000
max     75.000000   19.000000   19.000000   20.000000

Missing values in each column:
school        0
sex           0
age           0
address       0
famsize       0
Pstatus       0
Medu          0
Fedu          0
Mjob          0
Fjob          0
reason        0
guardian      0
traveltime    0
studytime     0
failures      0
schoolsup     0
famsup        0
paid          0
activities    0
nursery       0
higher        0
internet      0
romantic      0
famrel        0
freetime      0
goout         0
Dalc          0
Walc          0
health        0
absences      0
G1            0
G2            0
G3            0
dtype: int64
```

Data Preprocessing and Feature Encoding

```
# 1. Separate features and target
X = data.drop("G3", axis=1)  # All columns except G3
y = data["G3"]               # Target column

# 2. Encode categorical features
from sklearn.preprocessing import LabelEncoder

# Identify categorical columns
categorical_cols = X.select_dtypes(include='object').columns

# Apply Label Encoding to categorical columns
le = LabelEncoder()
for col in categorical_cols:
    X[col] = le.fit_transform(X[col])
```

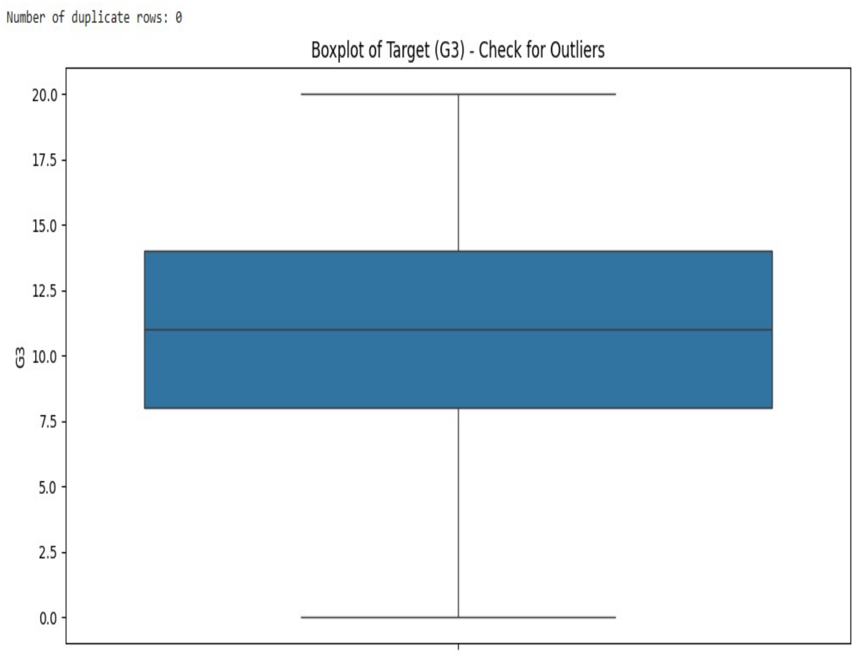
Feature Scaling and Data Cleaning

```
from sklearn.preprocessing import MinMaxScaler

numeric_cols = X.select_dtypes(include=['int64', 'float64']).columns
scaler = MinMaxScaler()
```

```
X[numeric_cols] = scaler.fit_transform(X[numeric_cols])

# 4. Check for duplicates
print("Number of duplicate rows:", data.duplicated().sum())
```

Outlier Detection

```
import matplotlib.pyplot as plt
import seaborn as sns

plt.figure(figsize=(12,6))
sns.boxplot(data=y)
plt.title("Boxplot of Target (G3) - Check for Outliers")
plt.show()
```

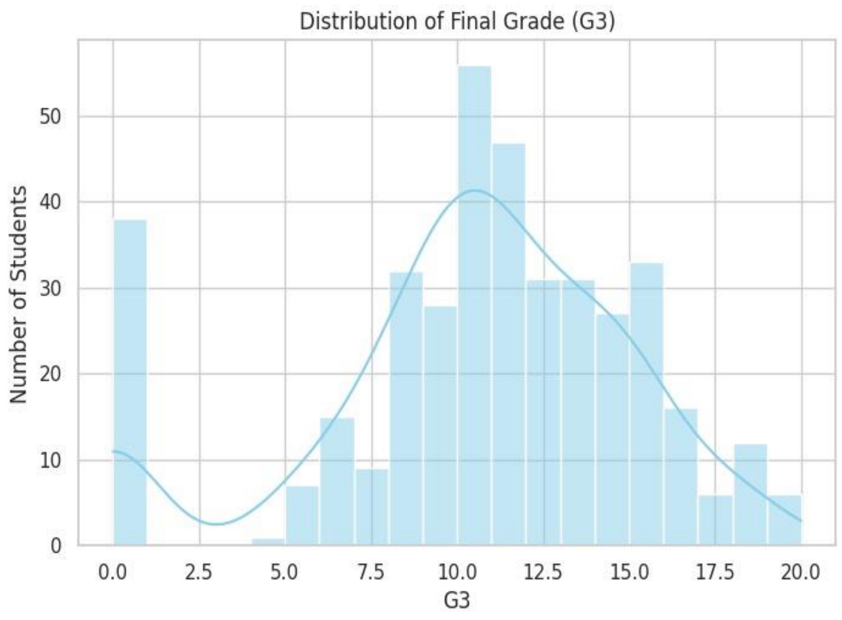


```
Number of duplicate rows: 0
```

Exploratory Data Analysis (EDA) and Visualization

Distribution of Final Grades

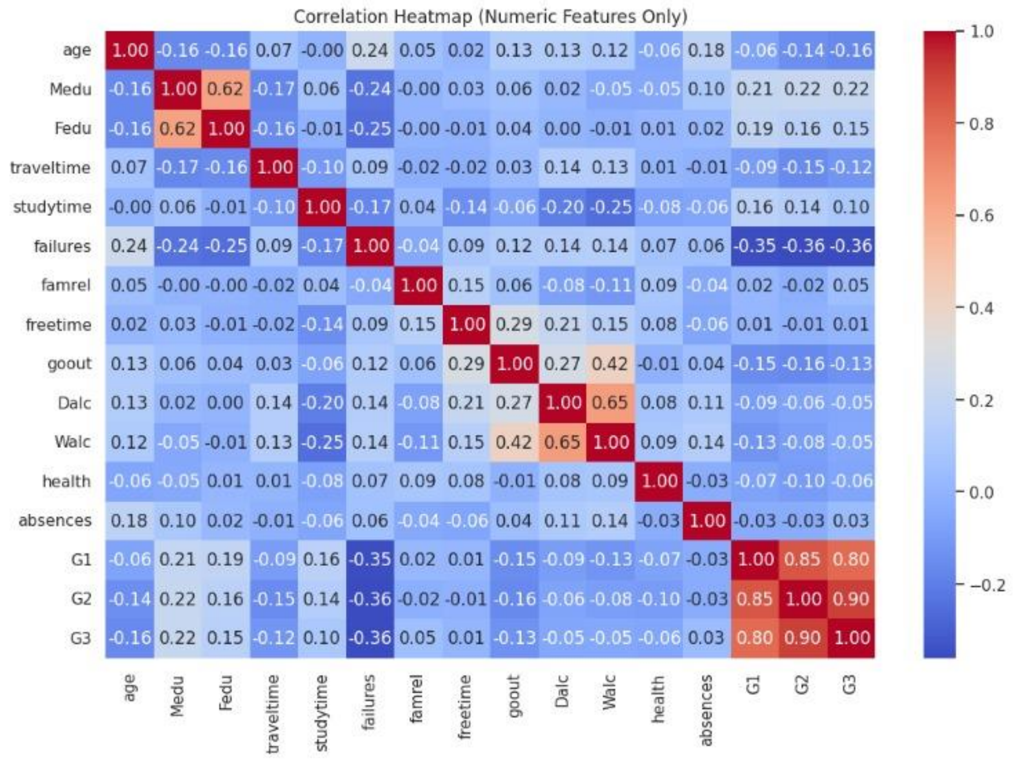
```
import matplotlib.pyplot as plt
import seaborn as sns

# Set a clean style for plots
sns.set(style="whitegrid")

# Plot the distribution of final grades (G3)
plt.figure(figsize=(8,5))
sns.histplot(y, bins=20, kde=True, color='skyblue')  # y is your target
variable G3
plt.title("Distribution of Final Grade (G3)")
plt.xlabel("G3")
plt.ylabel("Number of Students")
plt.show()
```



Correlation Heatmap

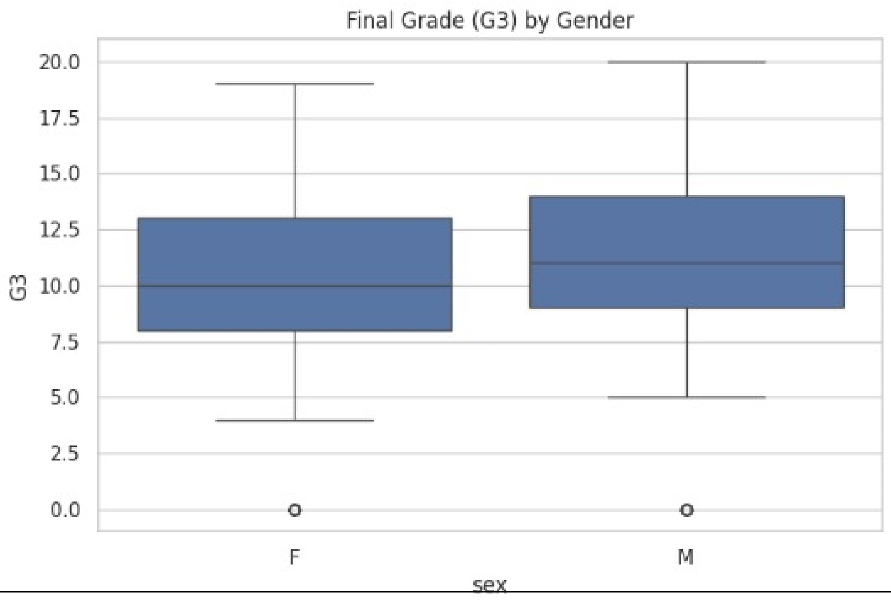
```
# Select only numeric columns
numeric_data = data.select_dtypes(include=['int64', 'float64'])

# Plot correlation heatmap
plt.figure(figsize=(12,8))
corr = numeric_data.corr()  # correlation between numeric features only
```

```
sns.heatmap(corr, annot=True, cmap='coolwarm', fmt=".2f")
plt.title("Correlation Heatmap (Numeric Features Only)")
plt.show()
```



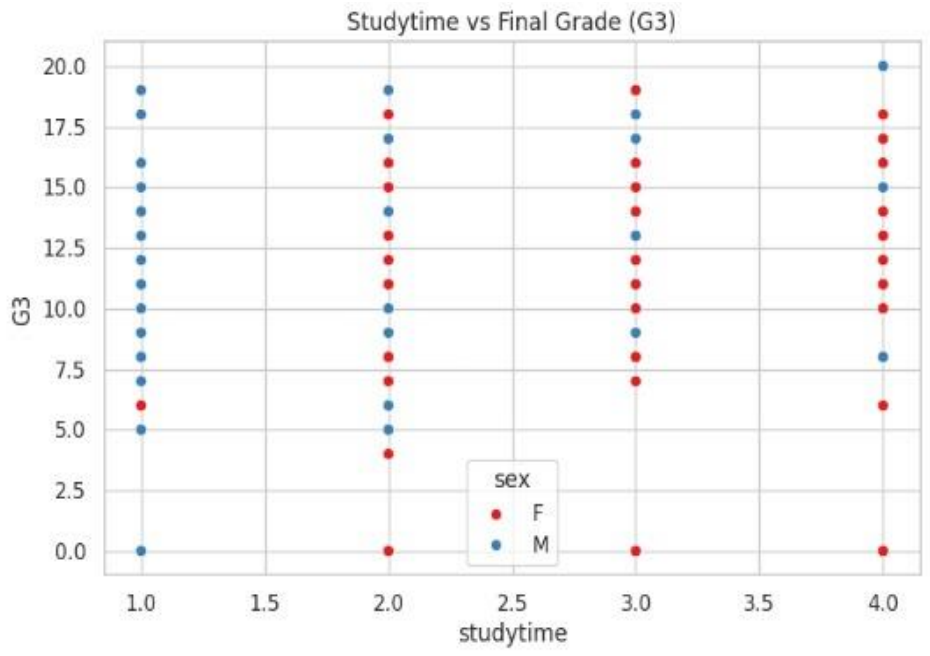
Final Grade by Gender

```
# Boxplot to compare final grades (G3) by student gender
plt.figure(figsize=(8,5))
sns.boxplot(x='sex', y='G3', data=data)
plt.title("Final Grade (G3) by Gender")
plt.show()
```



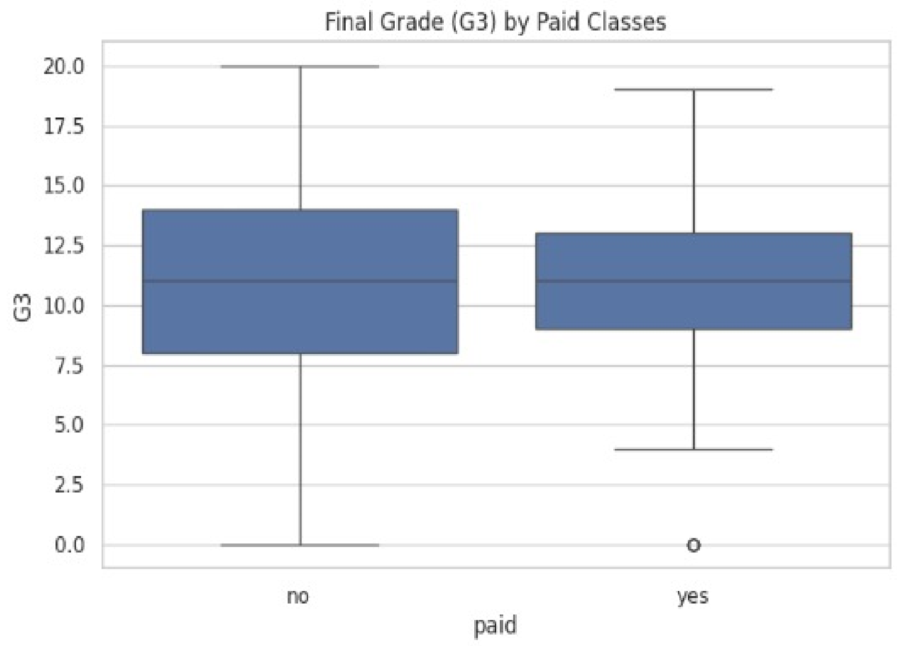
Studytime vs Final Grade

```
# Scatterplot to see relation between weekly study time and final grades
plt.figure(figsize=(8,5))
sns.scatterplot(x='studytime', y='G3', data=data, hue='sex',
palette='Set1')
plt.title("Studytime vs Final Grade (G3)")
plt.show()
```



Paid Classes vs Final Grade

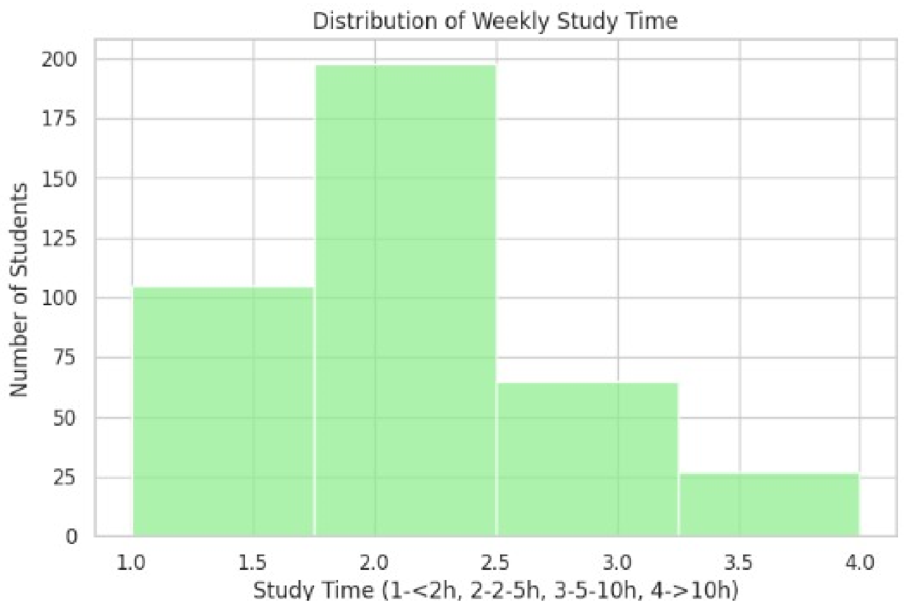
```
# Boxplot to analyze effect of paid extra classes on G3
plt.figure(figsize=(8,5))
sns.boxplot(x='paid', y='G3', data=data)
plt.title("Final Grade (G3) by Paid Classes")
plt.show()
```



Additional Visualizations

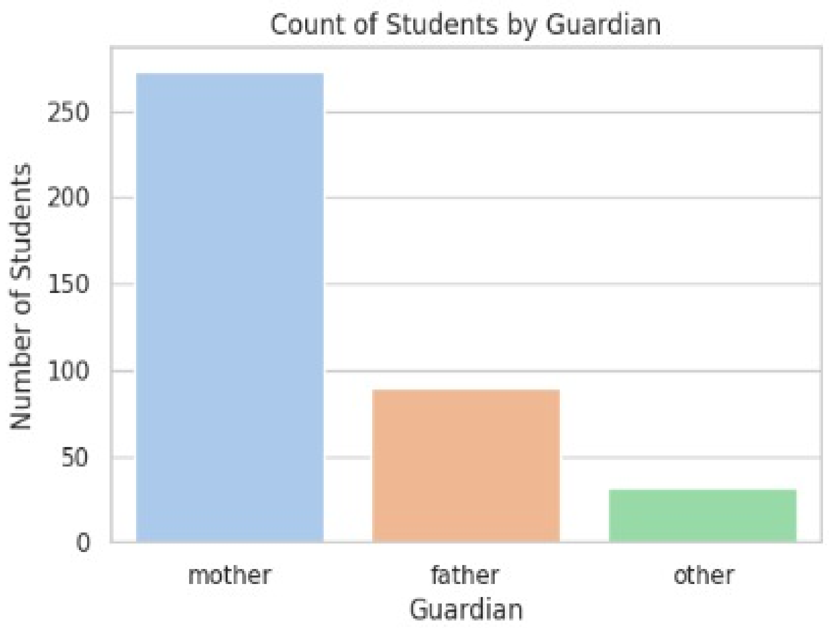
Distribution of Weekly Study Time

```
plt.figure(figsize=(8,5))
sns.histplot(data['studytime'], bins=4, kde=False, color='lightgreen')
plt.title("Distribution of Weekly Study Time")
plt.xlabel("Study Time (1-<2h, 2-2-5h, 3-5-10h, 4->10h)")
plt.ylabel("Number of Students")
plt.show()
```



Count of Students by Guardian

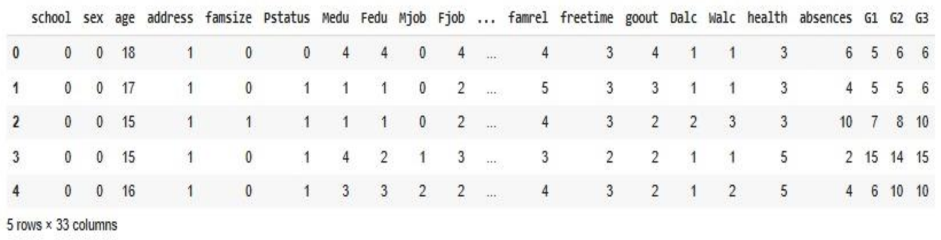
```
plt.figure(figsize=(6,4))
sns.countplot(x='guardian', data=data, palette='pastel')
plt.title("Count of Students by Guardian")
plt.xlabel("Guardian")
plt.ylabel("Number of Students")
plt.show()
```



Model Building and Evaluation (Machine Learning Models)

Label Encoding for Categorical Features

```
from sklearn.preprocessing import LabelEncoder

# List of categorical columns
categorical_cols =
['school','sex','address','famsize','Pstatus','Mjob','Fjob',
                    'reason','guardian','schoolsup','famsup','paid','activ
ities',
                    'nursery','higher','internet','romantic']

# Initialize LabelEncoder
le = LabelEncoder()

# Apply encoding
for col in categorical_cols:
    data[col] = le.fit_transform(data[col])

# Check first 5 rows after encoding
data.head()
```



	school	sex	age	address	famsize	Pstatus	Medu	Fedu	Mjob	Fjob	...	famrel	freetime	goout	Dalc	Walc	health	absences	G1	G2	G3
0	0	0	18	1	0	0	4	4	0	4	...	4	3	4	1	1	3	6	5	6	6
1	0	0	17	1	0	1	1	1	0	2	...	5	3	3	1	1	3	4	5	5	6
2	0	0	15	1	1	1	1	1	0	2	...	4	3	2	2	3	3	10	7	8	10
3	0	0	15	1	0	1	4	2	1	3	...	3	2	2	1	1	5	2	15	14	15
4	0	0	16	1	0	1	3	3	2	2	...	4	3	2	1	2	5	4	6	10	10

5 rows × 33 columns

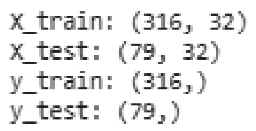
Splitting the Dataset

```
# Features (X) and target (y)
X = data.drop('G3', axis=1)
y = data['G3']

# Check shapes
print("Features shape:", X.shape)
print("Target shape:", y.shape)
```

```
  Features shape: (395, 32)
  Target shape: (395,)
```

```
from sklearn.model_selection import train_test_split
```

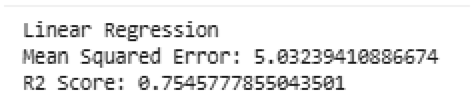
```python
# 80% training, 20% testing
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2,
random_state=42)

# Check shapes
print("X_train:", X_train.shape)
print("X_test:", X_test.shape)
print("y_train:", y_train.shape)
print("y_test:", y_test.shape)
```



```
X_train: (316, 32)
X_test: (79, 32)
y_train: (316,)
y_test: (79,)
```

Linear Regression Model

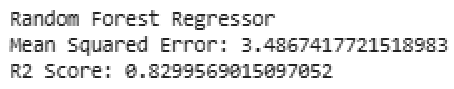
```python
from sklearn.linear_model import LinearRegression
from sklearn.metrics import mean_squared_error, r2_score

# Initialize the model
lr_model = LinearRegression()

# Train the model
lr_model.fit(X_train, y_train)

# Predict on test data
y_pred_lr = lr_model.predict(X_test)

# Evaluate the model
mse = mean_squared_error(y_test, y_pred_lr)
r2 = r2_score(y_test, y_pred_lr)

print("Linear Regression")
print("Mean Squared Error:", mse)
print("R2 Score:", r2)
```



```
Linear Regression
Mean Squared Error: 5.03239410886674
R2 Score: 0.7545777855043501
```

Random Forest Regressor

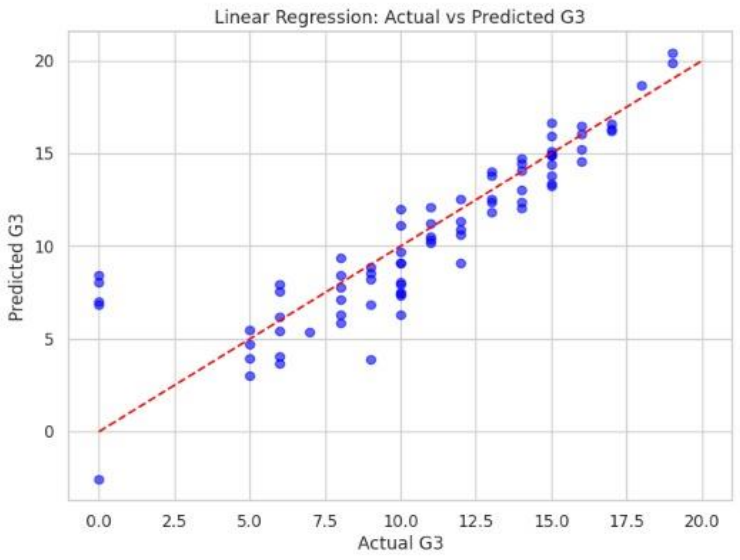
```
from sklearn.ensemble import RandomForestRegressor

# Initialize the model
rf_model = RandomForestRegressor(n_estimators=100, random_state=42)

# Train the model
rf_model.fit(X_train, y_train)

# Predict on test data
y_pred_rf = rf_model.predict(X_test)

# Evaluate the model
mse_rf = mean_squared_error(y_test, y_pred_rf)
r2_rf = r2_score(y_test, y_pred_rf)

print("Random Forest Regressor")
print("Mean Squared Error:", mse_rf)
print("R2 Score:", r2_rf)
```

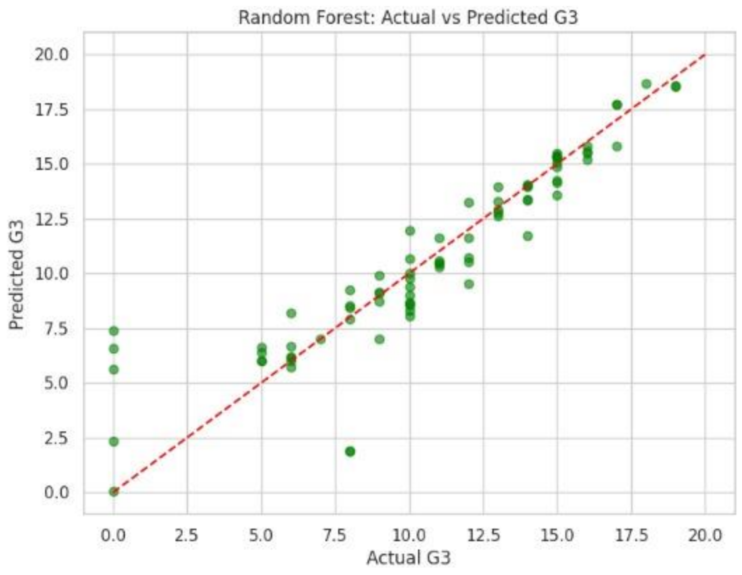


```
Random Forest Regressor
Mean Squared Error: 3.4867417721518983
R2 Score: 0.8299569015097052
```

Model Comparison – Actual vs Predicted Grades

```
plt.figure(figsize=(8,6))
plt.scatter(y_test, y_pred_lr, color='blue', alpha=0.6)
plt.plot([0, 20], [0, 20], color='red', linestyle='--')  # Perfect
prediction line
plt.xlabel("Actual G3")
plt.ylabel("Predicted G3")
plt.title("Linear Regression: Actual vs Predicted G3")
plt.show()



plt.figure(figsize=(8,6))
plt.scatter(y_test, y_pred_rf, color='green', alpha=0.6)
plt.plot([0, 20], [0, 20], color='red', linestyle='--')  # Perfect
prediction line
plt.xlabel("Actual G3")
plt.ylabel("Predicted G3")
plt.title("Random Forest: Actual vs Predicted G3")
plt.show()
```

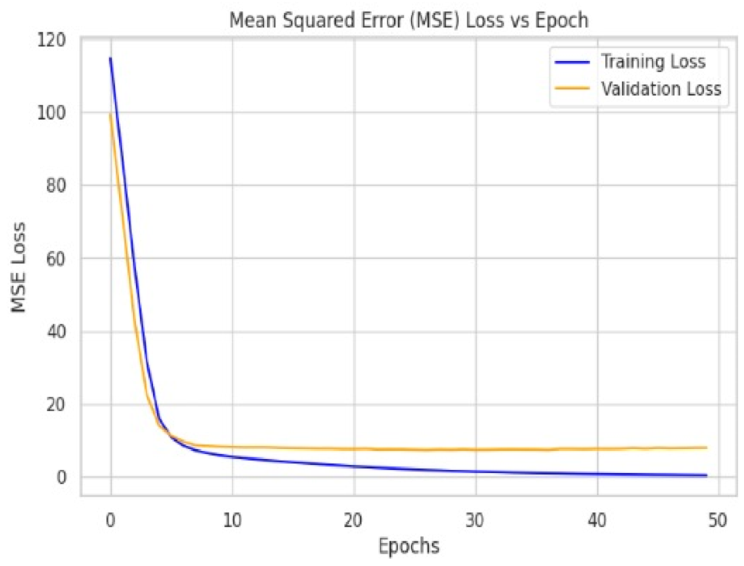


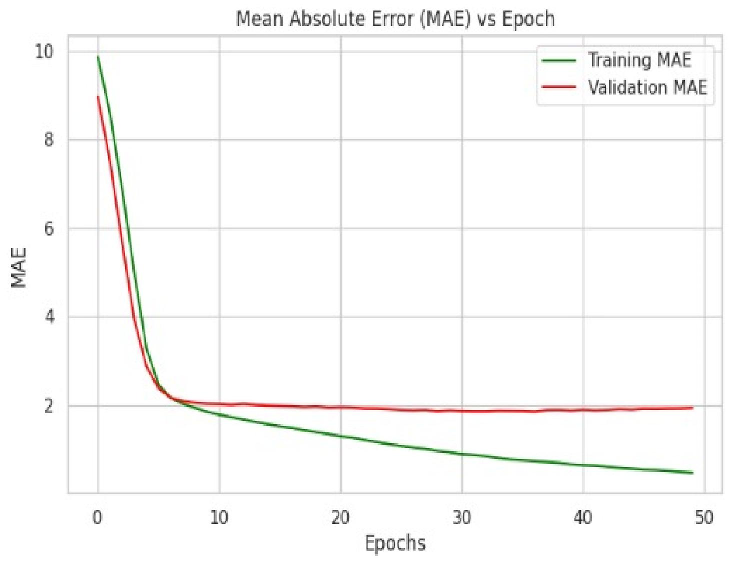


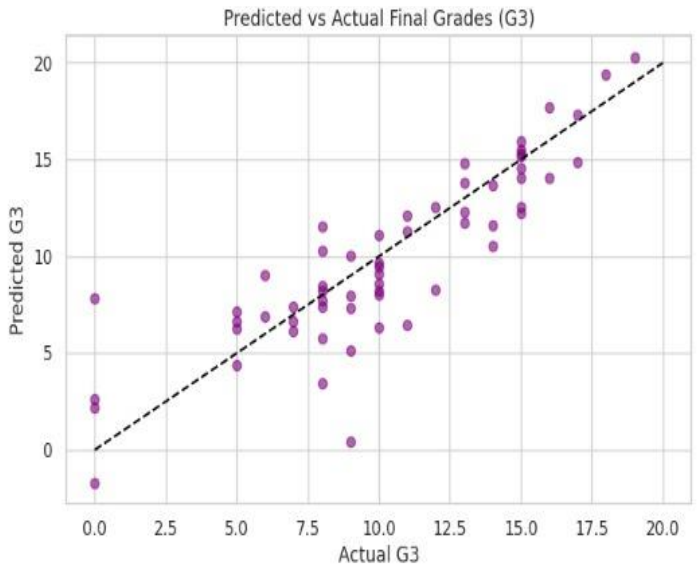
Result Visualization & Interpretation

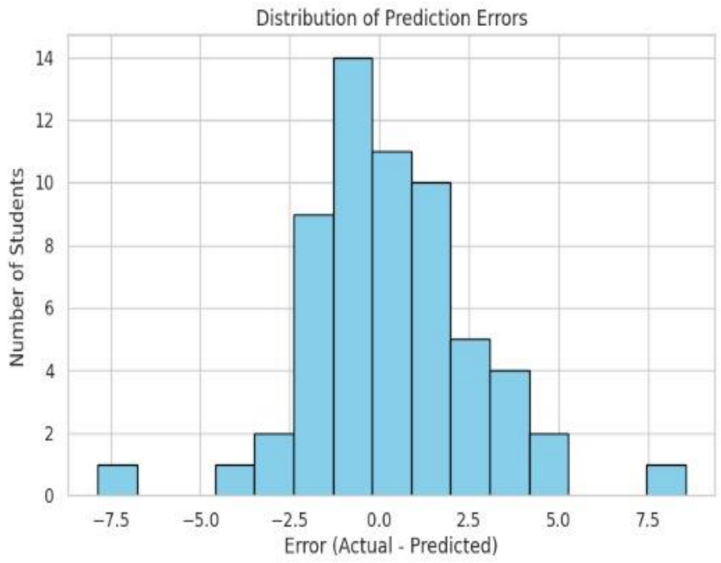
```
import matplotlib.pyplot as plt
import numpy as np

# ========================
# Step 1: Plot Loss vs Epoch
# ========================
plt.figure(figsize=(8,5))
plt.plot(history.history['loss'], label='Training Loss', color='blue')
plt.plot(history.history['val_loss'], label='Validation Loss',
color='orange')
plt.title("Mean Squared Error (MSE) Loss vs Epoch")
plt.xlabel("Epochs")
plt.ylabel("MSE Loss")
plt.legend()
plt.show()

# ========================
# Step 2: Plot MAE vs Epoch
# ========================
plt.figure(figsize=(8,5))
plt.plot(history.history['mae'], label='Training MAE', color='green')
plt.plot(history.history['val_mae'], label='Validation MAE', color='red')
plt.title("Mean Absolute Error (MAE) vs Epoch")
plt.xlabel("Epochs")
plt.ylabel("MAE")
plt.legend()
plt.show()

# ========================
# Step 3: Compare Predicted vs Actual G3
# ========================

# Predict on test set
y_pred = model.predict(X_test_scaled)

plt.figure(figsize=(8,5))
plt.scatter(y_test, y_pred, color='purple', alpha=0.6)
plt.plot([0, 20], [0, 20], color='black', linestyle='--')  # Perfect
prediction line
plt.title("Predicted vs Actual Final Grades (G3)")
plt.xlabel("Actual G3")
plt.ylabel("Predicted G3")
plt.show()
```

```
# ========================
# Step 4: Error Distribution
# ========================

errors = y_test - y_pred.flatten()
plt.figure(figsize=(8,5))
plt.hist(errors, bins=15, color='skyblue', edgecolor='black')
plt.title("Distribution of Prediction Errors")
plt.xlabel("Error (Actual - Predicted)")
plt.ylabel("Number of Students")
plt.show()
```






Predicted vs Actual Final Grades (G3)
Predicted G3
Actual G3
0.0
2.5
5.0
7.5
10.0
12.5
15.0
17.5
20.0
0
5
10
15
20


Distribution of Prediction Errors
Number of Students
Error (Actual - Predicted)
−7.5
−5.0
−2.5
0.0
2.5
5.0
7.5
0
2
4
6
8
10
12
14