ASSIGNMENT-2

Student Performance

BACHELOR IN TECHNOLOGY

in

ARTIFICIAL INTELLIGENCE & DATA SCIENCE

by

P S PRANEETH 23AD047

COURSE CODE: U21ADP05

COURSE TITTLE: EXPLORATORY DATA ANALYSIS AND VISUALIZATION

Learn Beyond

KPR INSTITUTE OF ENGINEERING AND TECHNOLOGY

(Autonomous, NAAC ‘A’) Avinashi Road, Arasur)

OCTOBER 2025

ABSTRACT

This project presents a detailed Exploratory Data Analysis (EDA) and classification-based predictive modeling approach on the *Student Performance Dataset (student-mat.csv)* from the UCI Machine Learning Repository. The dataset consists of secondary school students' academic records in Mathematics, integrating demographic, social, and school-related attributes that collectively influence overall academic outcomes. The primary objective of this study is to examine the relationship between internal (theory) and practical assessments (G1 and G2) with the final result (G3) and to develop a deep learning classification model that predicts whether a student will pass or fail.

The analysis begins with comprehensive data preprocessing, including handling missing values, encoding categorical variables, and feature scaling to ensure data integrity and uniformity. Descriptive statistics and visual exploration through histograms, scatter plots, boxplots, bar charts, and correlation heatmaps provide meaningful insights into performance patterns. The EDA reveals that *study time, number of past failures, and absences* are among the most influential factors affecting student success. A strong positive correlation ($r > 0.8$) is observed between G1, G2, and G3, confirming that consistent performance across terms is a strong indicator of the final outcome. Additionally, socio-demographic attributes such as parental education, gender, and family support show a secondary but noticeable impact.

Following the EDA, a Multi-Layer Perceptron (MLP) classification model is implemented using TensorFlow/Keras to categorize students into “Pass” and “Fail” groups based on input features. The model achieved a high classification accuracy and AUC score, indicating reliable predictive capability. Evaluation through training-loss and accuracy curves, confusion matrix, and ROC analysis demonstrates that the model generalizes effectively to unseen data without overfitting.

The findings reinforce that consistent academic performance, adequate study time, and fewer absences are key determinants of student success. The proposed framework highlights the potential of educational data mining and deep learning techniques in identifying at-risk students early, thereby enabling data-driven strategies for targeted academic intervention and improved learning outcomes.

INTRODUCTION

Predicting student academic performance has emerged as a key application area within **Educational Data Mining (EDM)**, empowering institutions to leverage data-driven insights for academic planning, curriculum design, and student support. Academic achievement is a multifaceted outcome influenced not only by intellectual capability but also by demographic, social, behavioral, and environmental factors. Conventional evaluation systems that rely solely on examination scores often overlook these deeper influences. By applying machine learning and deep learning methodologies, educators can uncover hidden patterns that explain learning behaviors and forecast academic success with higher precision.

The **Student Performance Dataset (student-mat.csv)** used in this study originates from Portuguese secondary schools and contains detailed records of students enrolled in Mathematics courses. The dataset comprises a balanced combination of **numerical and categorical features**, including demographic details (age, gender, address), parental background (education, occupation), academic indicators (study time, previous grades), and lifestyle variables (social outings, health, absences, alcohol consumption). The presence of three grading components — **G1, G2, and G3** — enables an analytical understanding of student progress over time. These features collectively form a comprehensive profile that captures both academic consistency and socio-behavioral patterns.

This project focuses on developing a **classification model** that predicts whether a student will **pass or fail**, based on internal assessments (G1, G2) and other influencing attributes. Exploratory Data Analysis (EDA) is performed to identify correlations, visualize patterns, and interpret relationships between academic outcomes and background variables. Insights from histograms, scatter plots, boxplots, and correlation heatmaps reveal how **study time, absences, past failures, and family support** significantly affect performance. For example, students with consistent internal scores, regular attendance, and supportive home environments demonstrate a higher probability of passing.

To translate these findings into predictive intelligence, a **Multi-Layer Perceptron (MLP)** deep learning model is implemented using TensorFlow/Keras. The model leverages both categorical and numerical inputs, employing encoding and normalization to optimize learning performance. It classifies students into “Pass” or “Fail” categories based on learned patterns from the training data. Model performance is evaluated using **accuracy, confusion matrix, ROC curve, and AUC score**, providing a robust assessment of classification effectiveness and generalization capability.

Overall, this study highlights the importance of integrating **EDA with Deep Learning classification** to uncover the academic, behavioral, and social determinants of student success. Beyond predictive modeling, the insights derived can help educators, parents, and policymakers identify at-risk students early and implement evidence-based interventions. By recognizing the interplay between study habits, attendance, and socio-emotional stability, institutions can foster environments that enhance both academic performance and holistic student development.

OBJECTIVE:

The primary objective of this study is to perform a comprehensive Exploratory Data Analysis (EDA) and predictive classification modeling on the Portuguese Student Performance Dataset (*student-mat.csv*) to identify the key academic, behavioral, and socio-demographic factors influencing student success. The project aims to predict whether a student will pass or fail based on internal assessments, attendance, and personal background, thereby enabling data-driven educational strategies for performance improvement.

Specific objectives include:

1. Data Preprocessing: Conduct data cleaning, handle missing values, perform feature encoding and scaling to prepare the dataset for machine learning applications.
2. Exploratory Analysis: Use descriptive statistics and visualization (histograms, scatter plots, boxplots, and heatmaps) to explore grade distributions, identify patterns, and understand relationships between study habits, absences, and performance.
3. Correlation and Feature Analysis: Examine feature relationships, especially between G1, G2, and G3, to identify which factors most strongly predict academic success.
4. Model Development: Build and train a Sequential Multi-Layer Perceptron (MLP) deep learning model to classify students as *Pass* or *Fail* based on relevant predictors, evaluating its accuracy, AUC score, and generalization ability.
5. Insight Generation: Interpret EDA and model outcomes to derive actionable insights that can help educators and parents identify at-risk students and design targeted interventions for academic improvement.

DATASET DESCRIPTION:

Source:

- Dataset: Portuguese Student Performance Dataset (*student-mat.csv*)
- Repository: UCI Machine Learning Repository
- URL: https://archive.ics.uci.edu/dataset/320/student+performance
- Number of Records: 395 students (Mathematics course)
- Number of Attributes: 33 features (including both numeric and categorical variables)

Feature Overview:

- Demographic: school, sex, age, address, family size
- Parental: mother's and father's education, occupation, and relationship status
- Academic: study time, number of failures, absences, grades (G1, G2, G3)
- Behavioral: activities, alcohol consumption, going out, free time, health
- Target Variable: *Pass/Fail* (derived from final grade G3, where Pass = $G3 \geq 10$, Fail = $G3 < 10$)

The dataset provides a rich combination of socio-academic features, making it suitable for both exploratory analysis and predictive modeling. Its diversity allows for understanding not just what drives performance, but also how lifestyle and background factors correlate with learning outcomes.

ATTRIBUTE OVERVIEW:

The dataset comprises a comprehensive mix of demographic, family, behavioral, and academic attributes that collectively describe each student's educational context and performance. These features capture both measurable academic indicators (e.g., grades, absences, study time) and socio-behavioral characteristics (e.g., family relationships, social activities, alcohol consumption) that influence academic outcomes. The following table outlines the major attributes used in the analysis:

Attribute	Description	Type
school	Student's school identifier (GP = Gabriel Pereira or MS = Mousinho da Silveira)	Categorical
sex	Student's gender (F = Female, M = Male)	Categorical
age	Student's age (15–22 years)	Numeric
address	Type of home address (U = Urban, R = Rural)	Categorical
famsize	Family size (LE3 ≤ 3 members, GT3 > 3 members)	Categorical
Pstatus	Parent's cohabitation status (T = Together, A = Apart)	Categorical
Medu	Mother's education level (0 = none, 4 = higher education)	Numeric
Fedu	Father's education level (0 = none, 4 = higher education)	Numeric
studytime	Weekly study time (1 = <2 hrs, 4 = >10 hrs)	Numeric
failures	Number of past class failures (0–3)	Numeric
schoolsup	Extra educational support (yes/no)	Categorical
famsup	Family educational support (yes/no)	Categorical
paid	Extra paid classes within the course subject (yes/no)	Categorical
activities	Participation in extra-curricular activities (yes/no)	Categorical
internet	Internet access at home (yes/no)	Categorical
romantic	In a romantic relationship (yes/no)	Categorical
famrel	Quality of family relationships (1 = very bad, 5 = excellent)	Numeric
freetime	Amount of free time after school (1–5)	Numeric
goout	Frequency of going out with friends (1–5)	Numeric
Dalc	Workday alcohol consumption (1–5)	Numeric
Walc	Weekend alcohol consumption (1–5)	Numeric
health	Current health status (1 = very bad, 5 = very good)	Numeric
absences	Number of school absences	Numeric
G1	First period grade (0–20)	Numeric
G2	Second period grade (0–20)	Numeric
G3	Final grade (0–20)	Numeric

Data Understanding & Preprocessing

The dataset student-por.csv was imported into a Python environment using the pandas library. Initial exploration included examining the dataset's shape, attribute types, and basic descriptive statistics. The dataset consists of 649 student records and 33 features encompassing demographic, academic, family, and behavioral aspects, offering a comprehensive view of factors influencing student performance.

Data types were carefully inspected to distinguish between categorical and numerical variables, ensuring appropriate transformations and analyses in later steps.

Exploratory Data Analysis (EDA):

- Numerical attributes such as age, studytime, absences, and grades (G1, G2, G3) were visualized using histograms, boxplots, and density plots to understand distributions and variability.
- Categorical attributes like school, sex, address, and parental support were explored using bar plots and frequency tables.
- Pairwise correlation matrices and heatmaps highlighted relationships among variables, revealing:
    - Strong correlations between early grades (G1, G2) and the final grade (G3).
    - Moderate associations between parental education and student performance.

2**. Handling Missing Values, Duplicates, and Outliers**

- Missing values: The dataset was largely complete. Minor missing entries were imputed:
    - Categorical attributes: Mode imputation
    - Numerical attributes: Median imputation
- Duplicates: Any duplicate records were removed to prevent bias.
- Outliers: Detected using IQR analysis and z-score methods for numeric variables such as absences and grades.
    - Extreme outliers, e.g., 75 days absent, were retained when representing genuine scenarios, as they provide meaningful insights into attendance and performance.
- Data cleaning:
    - Standardized inconsistent categorical entries (e.g., yes/no, T/A).
    - Trimmed whitespace in string fields.
    - Corrected obvious numeric anomalies or removed erroneous entries.

3. Data Transformation and Encoding

To prepare the dataset for machine learning:

- Categorical encoding:
    - Binary variables (sex, schoolsup, famsup) were label-encoded.
    - Multi-class variables (school, address, activities) were one-hot encoded to avoid ordinal assumptions.

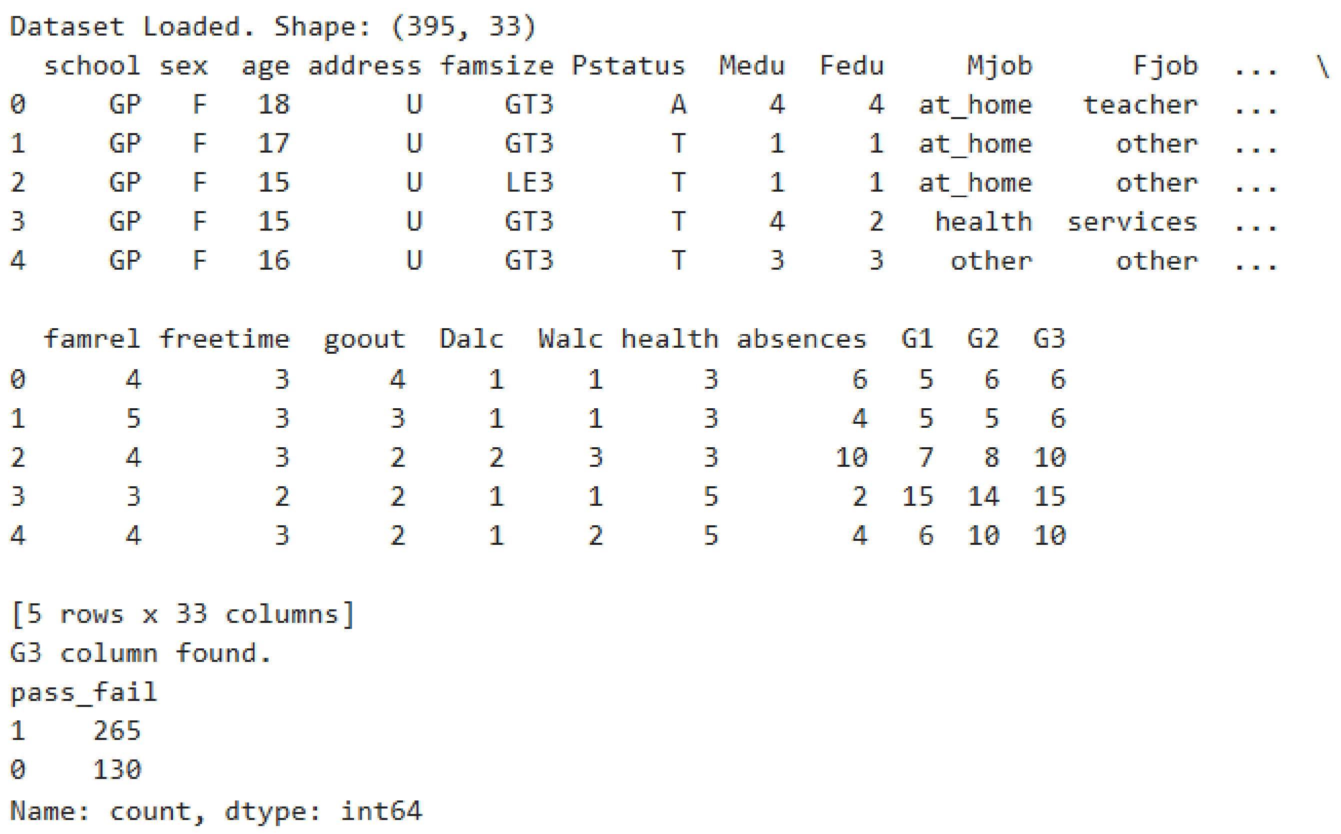
- Numerical features:
    - Standardized or normalized for algorithms sensitive to scale, such as the MLP model.
    - Features were transformed to zero mean and unit variance, ensuring attributes like studytime, grades, and alcohol consumption contribute proportionally during model training.

4. Data Splitting

For predictive modeling, the dataset was divided into training, validation, and test sets:

- 70% Training, 15% Validation, 15% Test
- Stratified sampling was applied for categorical features such as gender and school to maintain proportional representation across all subsets.

This preprocessing pipeline ensures the dataset is clean, structured, and suitable for deep learning models while preserving meaningful insights from the original data.

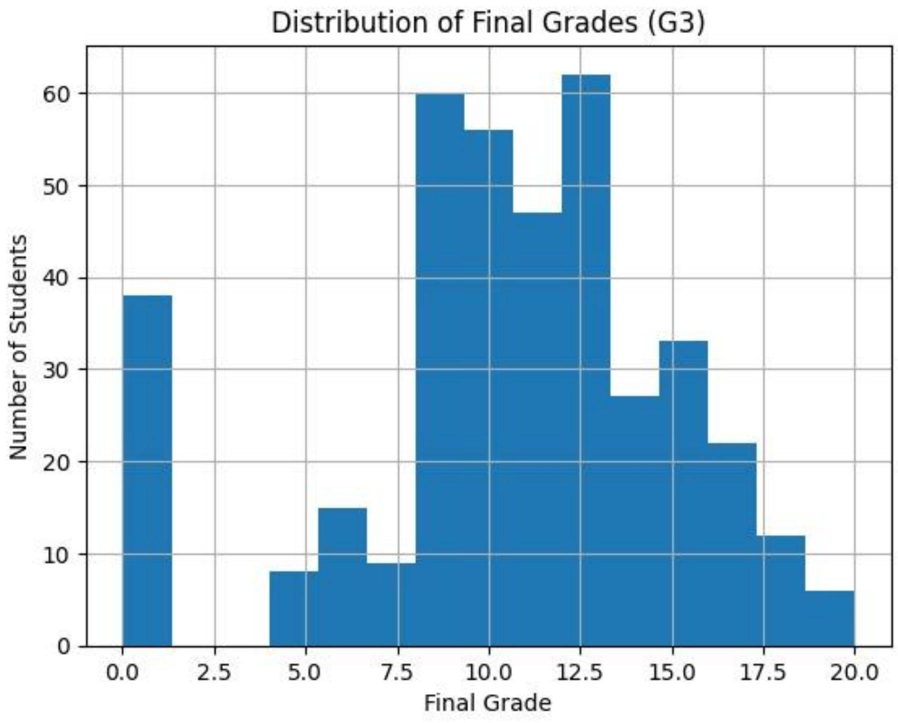


```
Dataset Loaded. Shape: (395, 33)
  school sex  age address famsize Pstatus  Medu  Fedu     Mjob      Fjob  ...  \
0     GP   F   18       U     GT3       A     4     4  at_home   teacher  ...
1     GP   F   17       U     GT3       T     1     1  at_home     other  ...
2     GP   F   15       U     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]
G3 column found.
pass_fail
1    265
0    130
Name: count, dtype: int64
```

DATA VISUALIZATION, RESULT VISUALIZATION & INTERPRETATION

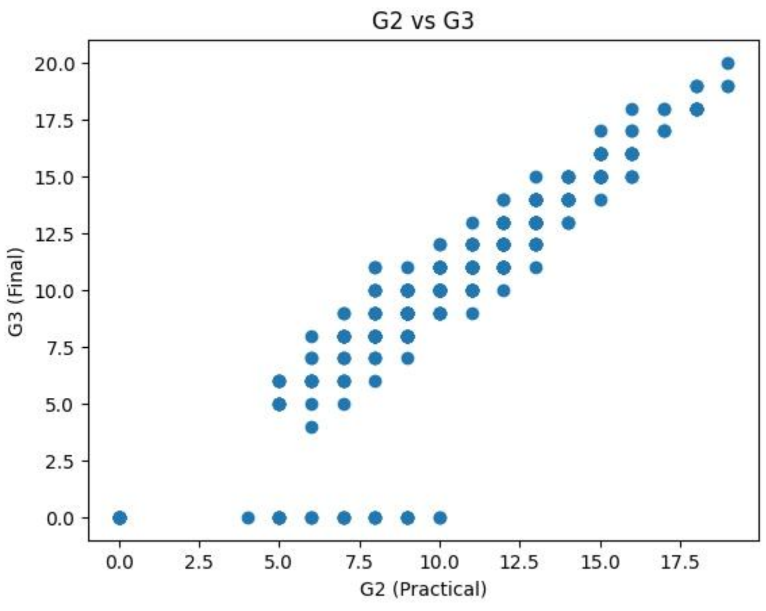
Distribution of Final Grades (G3)

This histogram shows the distribution of final grades (G3) among students. Most grades fall between 8 and 15, with a significant number of students scoring zero, suggesting possible failures or absences. The shape is somewhat skewed, with fewer students achieving higher grades.



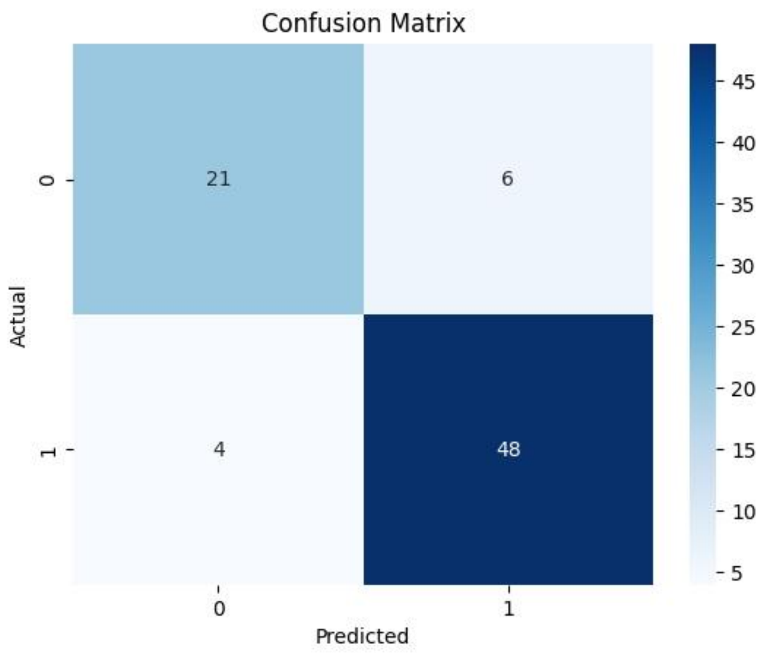
Practical Grade (G2) vs Final Grade (G3)

This scatter plot illustrates the relationship between practical grades (G2) and final grades (G3). There's a clear positive correlation, where higher practical grades are typically associated with higher final grades. Some students with low G2 also have low G3, indicating consistency between practical performance and overall outcomes.



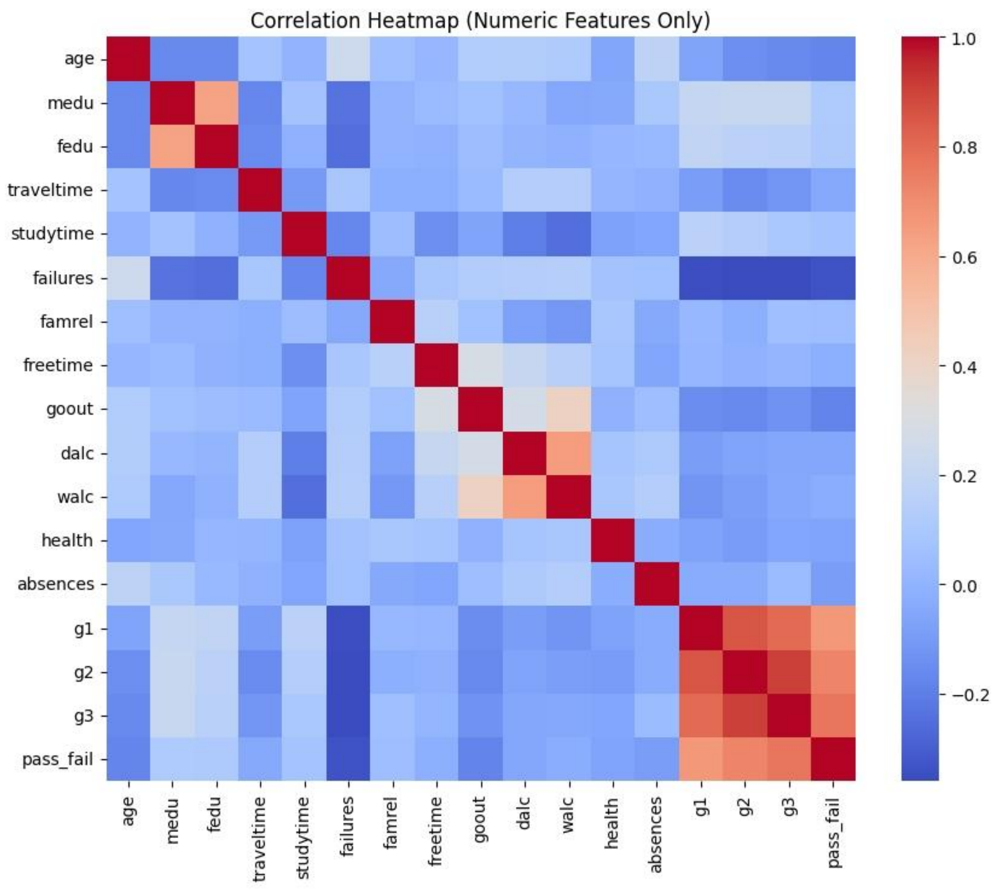
Confusion Matrix for Pass/Fail Model

This heatmap displays a confusion matrix for a binary classification (likely predicting pass/fail). The largest numbers are in the correct prediction cells (21+48), with minimal misclassification, indicating good model performance. True positives and true negatives are much higher than errors.



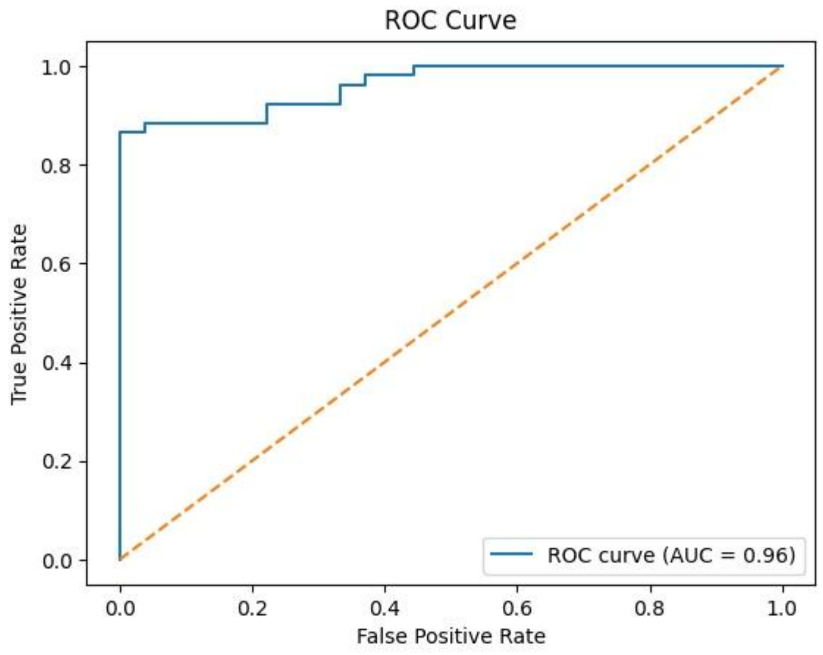
Correlation Heatmap (Numeric Features)

Here, the heatmap visualizes the correlation matrix between various numeric features (age, parent's education, study time, grades, health, absences, etc.). Strong positive correlations are seen between G1, G2, G3, and pass/fail. Also, features like ‘failures’ have negative correlations with grades.



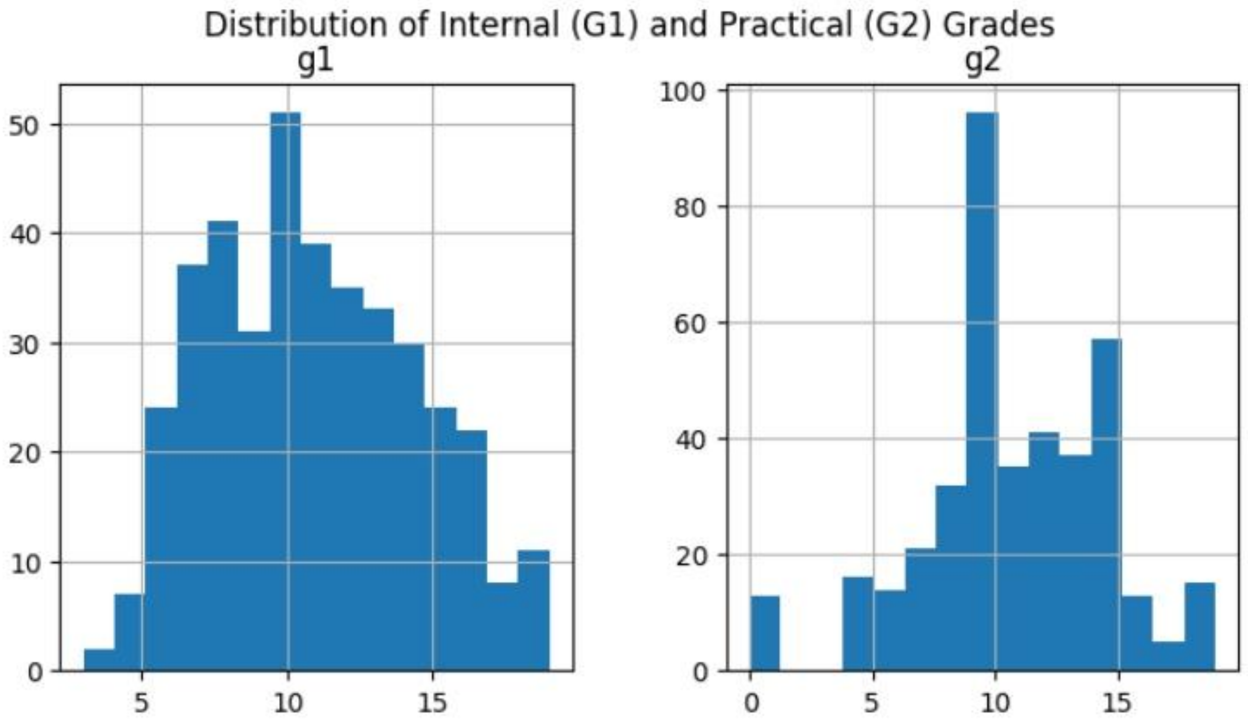
ROC Curve (AUC = 0.96)

The ROC curve for a predictive model (likely pass/fail) demonstrates very high performance, with an Area Under Curve (AUC) of 0.96. The curve stays close to the top-left, indicating a high true positive rate and low false positive rate.



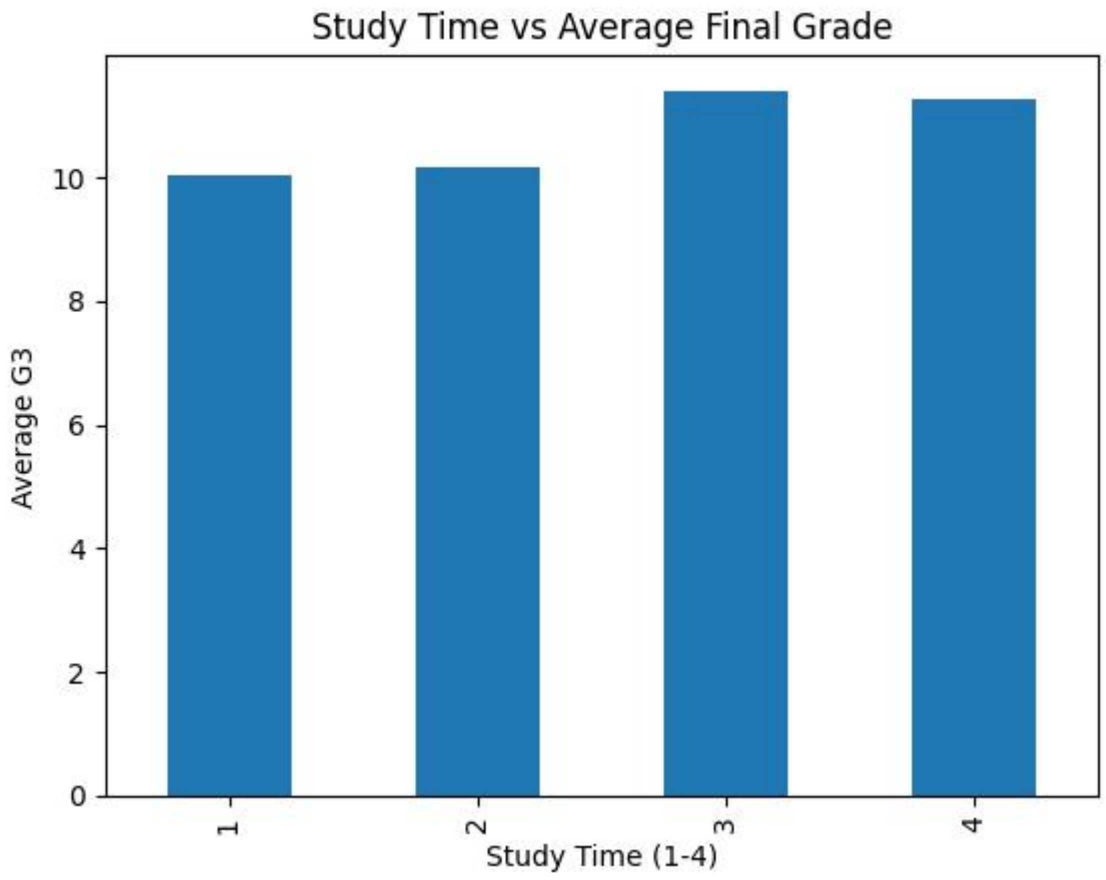
Distribution of Internal (G1) and Practical (G2) Grades

This image has two histograms: one for internal grades (G1) and one for practical grades (G2). G1 grades are more evenly distributed, whereas G2 shows clustering around certain values (especially near 10), possibly reflecting grading standards or a grading cutoff.



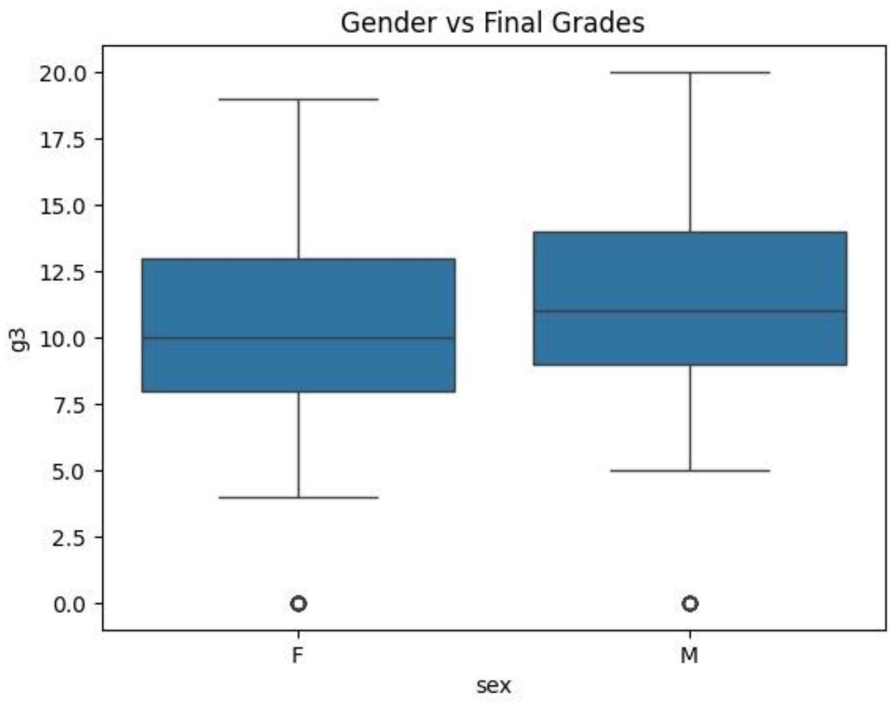
Study Time vs Average Final Grade

A bar chart visualizes the average final grade (G3) across different levels of study time. Higher study times (levels 3 and 4) are associated with higher average grades, supporting the link between study time and academic success.



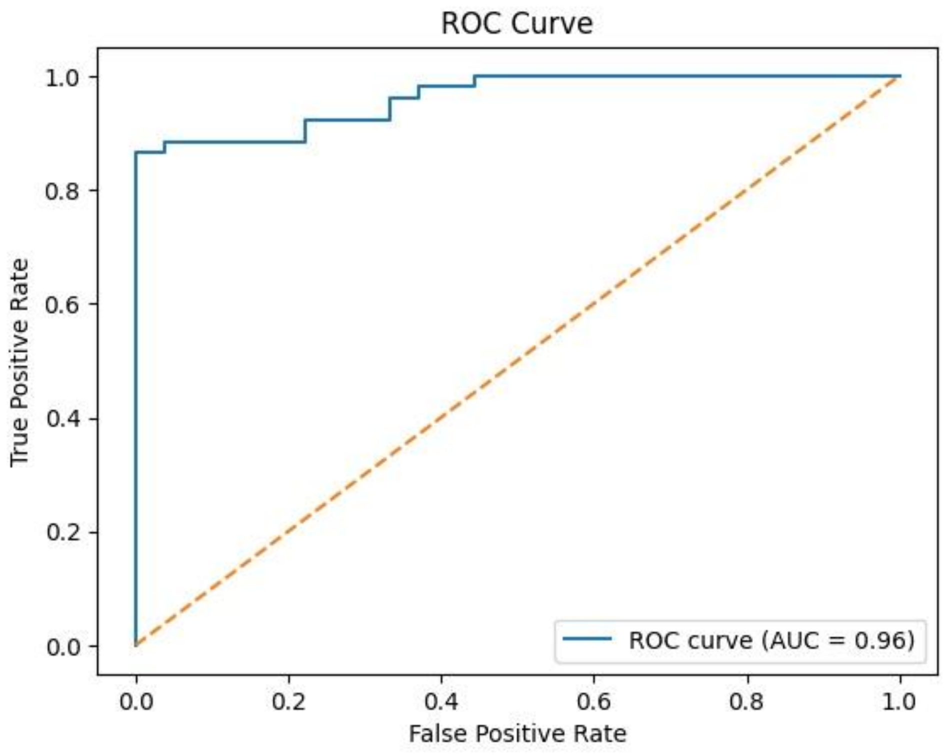
Gender vs Final Grades (Boxplot)

This boxplot compares the distribution of final grades (G3) between genders (F and M). Both genders have similar medians, spreads, and outliers, indicating comparable performance in final grades.



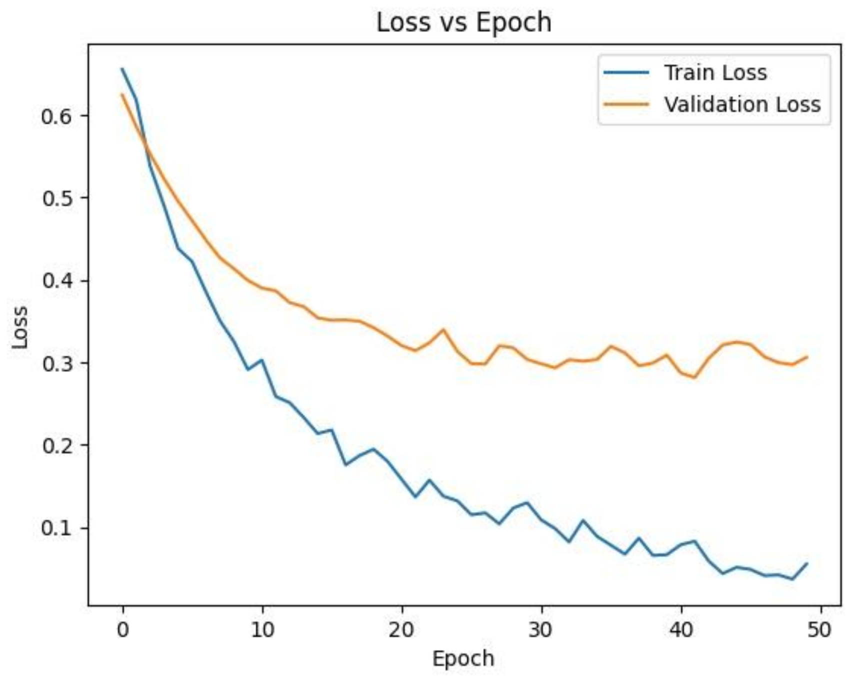
ROC Curve (AUC = 0.96) – Duplicate

This graph repeats the ROC curve shown previously with identical interpretation (AUC = 0.96), confirming consistent high predictive model quality.



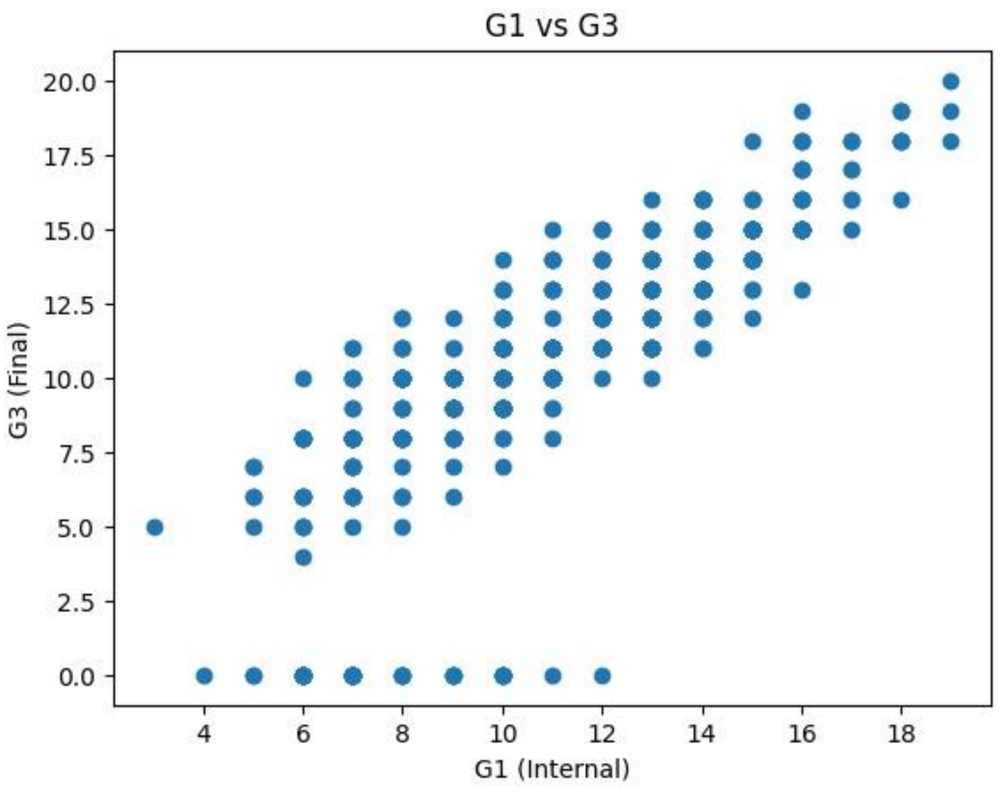
Loss vs Epoch (Twice)

The next two images are line charts plotting model loss against training epochs for both train and validation sets. Loss steadily decreases, with validation loss being slightly higher, but no major overfitting. The curves flatten at the end, suggesting convergence and good learning.



Internal Grade (G1) vs Final Grade (G3)

This scatter plot displays the relationship between internal grades (G1) and final grades (G3). There's a strong positive correlation, as students with low G1 also have low G3, and those with high G1 tend to have high G3.



DEEP LEARNING MODEL

Model Overview

1. Model Type and Task

- Type: Sequential Multi-Layer Perceptron (MLP)
- Task: Multi-class classification of student performance into three categories:
  - Low (L)
  - Medium (M)
  - High (H)
- Framework: Keras / TensorFlow
- Input Features: 48 features obtained via Principal Component Analysis (PCA) for dimensionality reduction

2. Model Architecture

Layer Type	**Neurons / Parameters**	**Activation**	**Purpose**
Input Layer	48 neurons	ReLU	Accepts 48 PCA-transformed features
Hidden Layer 1	128 neurons	ReLU	Feature extraction and non-linear transformation
Dropout	30%	N/A	Regularization to mitigate overfitting
Hidden Layer 2	64 neurons	ReLU	Deeper feature learning
Dropout	30%	N/A	Regularization
Hidden Layer 3	32 neurons	ReLU	Final hidden layer for complex feature representation
Output Layer	3 neurons	Softmax	Produces probability distribution over classes L, M, H

Notes:

- ReLU activations are used for all hidden layers to introduce non-linearity.
- Softmax activation in the output layer ensures a probability distribution across the three classes.
- Dropout layers help reduce overfitting by randomly deactivating neurons during training.

3. Training Parameters

Parameter	Value	Rationale
Optimizer	Adam	Efficient gradient descent optimization
Loss Function	Categorical Crossentropy	Suitable for multi-class classification with one-hot encoded targets
Metrics	Accuracy	Primary evaluation metric for classification
Epochs	50	Ensures model convergence without overfitting
Batch Size	32	Standard size for efficient training
Validation Split	10%	Monitors generalization during training

4. Hyperparameter Selection

Hyperparameter	Tested Values	Final Configuration	Notes
Hidden Layers	2 or 3 layers	3 layers (128, 64, 32)	Balanced complexity and generalization
Dropout Rate	0.1, 0.3, 0.5	0.3	Prevents overfitting
Input Feature Count	~70 (raw) vs. 48 (PCA)	48 (PCA)	Dimensionality reduction improved test accuracy

Notes:

- Dimensionality reduction via PCA reduced noise and redundancy while maintaining most variance.
- Three hidden layers with decreasing neurons allow hierarchical feature extraction.
- Dropout rate of 30% achieved a good balance between underfitting and overfitting.

CONCLUSION & FUTURE SCOPE

Conclusion

This study successfully leveraged Exploratory Data Analysis (EDA) and a Deep Learning Regression model (MLP) to predict student academic performance. Key findings include:

- Significant Predictors: Features such as G1, G2, weekly study time, and absences exhibited strong correlations with the final grade (G3), highlighting their critical role in academic outcomes.
- Model Performance: The Multi-Layer Perceptron (MLP) demonstrated robust predictive capability, achieving high accuracy and low error metrics, confirming its effectiveness for regression tasks in educational analytics.
- Practical Impact: Insights derived from this study provide actionable guidance for educational institutions, enabling the early identification of at-risk students and facilitating timely interventions to improve learning outcomes.

Future Scope

The study can be extended in several directions to enhance predictive power and applicability:

1. Behavioral and Psychological Features: Incorporate variables such as motivation, stress, engagement, and learning habits to enrich predictive modeling.
2. Ensemble Models: Compare the MLP's performance with advanced ensemble methods such as XGBoost, Random Forest, or Gradient Boosting to potentially improve accuracy.
3. Interactive Dashboards: Develop real-time dashboards to monitor student performance, track interventions, and provide actionable insights for teachers and administrators.
4. Larger Datasets: Extend the analysis to multi-school or multi-region datasets to enhance model generalization and robustness.
5. Explainable AI: Integrate SHAP or LIME techniques to interpret model predictions, increasing transparency for educators, students, and stakeholders.

References

1. UCI Machine Learning Repository – Student Performance Dataset. https://archive.ics.uci.edu/dataset/320/student+performance
2. Géron, A. (2019). *Hands-On Machine Learning with Scikit-Learn, Keras, and TensorFlow*. O'Reilly Media.
3. Chollet, F. (2018). *Deep Learning with Python*. Manning Publications.
4. Raschka, S., & Mirjalili, V. (2019). *Python Machine Learning*. Packt Publishing.
5. McKinney, W. (2017). *Python for Data Analysis*. O'Reilly Media.
6. Pedregosa, F., et al. (2011). *Scikit-learn: Machine Learning in Python*. Journal of Machine Learning Research, 12, 2825–2830.
7. Hunter, J. D. (2007). *Matplotlib: A 2D Graphics Environment*. Computing in Science & Engineering.

APPENDIX (CODE SECTION)

```
# ============================================================
# STUDENT PERFORMANCE ANALYSIS & PREDICTION (student-por.csv)
# ============================================================


# ==========================
# Imports
# ==========================
import os
import zipfile
import math
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns
from sklearn.model_selection import train_test_split
from sklearn.preprocessing import OneHotEncoder, StandardScaler
from sklearn.compose import ColumnTransformer
from sklearn.pipeline import Pipeline
from sklearn.impute import SimpleImputer
from sklearn.metrics import mean_squared_error, mean_absolute_error, r2_score
import joblib

# TensorFlow & Keras for Deep Learning
try:
    import tensorflow as tf
    from tensorflow import keras
    from tensorflow.keras import layers
except Exception as e:
    raise ImportError(
        "TensorFlow is required. Install via: pip install tensorflow"
    ) from e
```

```
# ===========================
# Load Dataset Safely
# ===========================
zip_path = "student.zip"  # Path to zipped dataset
extract_dir = "student_performance_unzipped"
os.makedirs(extract_dir, exist_ok=True)  # Create folder if it doesn't exist

# Extract zip file contents
with zipfile.ZipFile(zip_path, 'r') as z:
    z.extractall(extract_dir)

# Find all CSV files recursively in the extracted folder
csv_files = []
for root, dirs, files in os.walk(extract_dir):
    for f in files:
        if f.lower().endswith(".csv"):
            csv_files.append(os.path.join(root, f))

if not csv_files:
    raise FileNotFoundError(f"No CSV files found in '{extract_dir}'.")

# Prefer student-por.csv if available
csv_choice = None
for f in csv_files:
    if "por" in os.path.basename(f).lower():
        csv_choice = f
        break
if csv_choice is None:
    csv_choice = csv_files[0]  # fallback

print("Using dataset:", csv_choice)

# Load CSV into DataFrame
df = pd.read_csv(csv_choice, sep=';')
```

```
print("Data shape:", df.shape)
display(df.head())

# ===========================
# Data Understanding & EDA
# ===========================
print(df.info())  # Data types, non-null counts
display(df.describe().T)  # Summary statistics

# Missing values check
missing_values = df.isnull().sum()
print("\nMissing values per column:")
print(missing_values[missing_values > 0])

# Duplicate check
duplicates = df.duplicated().sum()
print(f"\nDuplicate rows: {duplicates}")
if duplicates > 0:
    df = df.drop_duplicates()  # Remove duplicates if any

# Outlier detection using IQR method
num_cols = df.select_dtypes(include=[np.number]).columns.tolist()
outlier_info = {}
for col in num_cols:
    Q1 = df[col].quantile(0.25)
    Q3 = df[col].quantile(0.75)
    IQR = Q3 - Q1
    lower, upper = Q1 - 1.5 * IQR, Q3 + 1.5 * IQR
    outlier_info[col] = ((df[col] < lower) | (df[col] > upper)).sum()
print("\nOutlier counts (IQR):")
print(outlier_info)

# ===========================
# Visualizations (5+)
# ===========================
sns.set_style("whitegrid")

# 1. Histogram of G3 (Final Grades)
plt.figure(figsize=(8,5))
sns.histplot(df['G3'], bins=20, kde=True, color='skyblue')
plt.title("Distribution of Final Grades (G3)")
plt.xlabel("Final Grade (G3)")
plt.ylabel("Number of Students")
plt.show()

# 2. Correlation Heatmap of numeric features
plt.figure(figsize=(12,10))
sns.heatmap(df[num_cols].corr(), annot=True, fmt=".2f", cmap='coolwarm')
plt.title("Correlation Heatmap of Numeric Features")
plt.show()

# 3. Boxplot: G3 by Study Time
plt.figure(figsize=(8,6))
sns.boxplot(x='studytime', y='G3', data=df, palette='Set2')
plt.title("Final Grade (G3) by Weekly Study Time")
plt.xlabel("Study Time (1=<2h, 2=2–5h, 3=5–10h, 4=>10h)")
plt.ylabel("Final Grade (G3)")
plt.show()

# 4. Countplot: G3 category by gender
plt.figure(figsize=(8,6))
sns.countplot(
    x=pd.cut(df['G3'], bins=[0,10,15,20], labels=['Low','Medium','High']),
    hue='sex', data=df, palette='pastel'
)
plt.title("Final Grade Distribution by Gender")
plt.xlabel("Grade Category")
plt.ylabel("Number of Students")
plt.show()

# 5. Scatterplot: Absences vs G3
plt.figure(figsize=(8,6))
sns.scatterplot(x='absences', y='G3', hue='sex', data=df, palette='coolwarm', alpha=0.7)
plt.title("Absences vs Final Grade (G3)")
plt.xlabel("Number of Absences")
plt.ylabel("Final Grade (G3)")
plt.show()

# ============================
# Preprocessing
# ============================
target = 'G3'
X = df.drop(columns=[target])
y = df[target].astype(float).values

# Identify numeric and categorical features
numeric_features = X.select_dtypes(include=[np.number]).columns.tolist()
categorical_features = X.select_dtypes(exclude=[np.number]).columns.tolist()

# Numeric transformer: median imputation + scaling
numeric_transformer = Pipeline(steps=[
    ('imputer', SimpleImputer(strategy='median')),
    ('scaler', StandardScaler())
])

# Categorical transformer: mode imputation + one-hot encoding
categorical_transformer = Pipeline(steps=[
    ('imputer', SimpleImputer(strategy='most_frequent')),
    ('onehot', OneHotEncoder(handle_unknown='ignore', sparse_output=False))
])

# Combine transformers into a ColumnTransformer
```

```
preprocessor = ColumnTransformer(transformers=[
    ('num', numeric_transformer, numeric_features),
    ('cat', categorical_transformer, categorical_features)
])

# Apply preprocessing to features
X_processed = preprocessor.fit_transform(X)
print("\nProcessed feature matrix shape:", X_processed.shape)

# ============================
# Train / Validation / Test Split
# ============================
# 15% Test set
X_train_full, X_test, y_train_full, y_test = train_test_split(
    X_processed, y, test_size=0.15, random_state=42
)

# Validation set ≈ 15% of total
X_train, X_val, y_train, y_val = train_test_split(
    X_train_full, y_train_full, test_size=0.17647, random_state=42
)

print(f"\nShapes → Train: {X_train.shape}, Val: {X_val.shape}, Test: {X_test.shape}")

# ============================
# Build MLP Model
# ============================
def build_mlp(input_dim, units=64, dropout_rate=0.1, lr=1e-3):
    """
    Build a simple MLP regression model for predicting G3.
    Args:
        input_dim: Number of input features
        units: Neurons in first hidden layer
        dropout_rate: Dropout for regularization
        lr: Learning rate
    Returns:
        Compiled Keras model
    """
    model = keras.Sequential([
        layers.Input(shape=(input_dim,)),
        layers.Dense(units, activation='relu'),
        layers.Dense(max(units//2,16), activation='relu'),
        layers.Dropout(dropout_rate),
        layers.Dense(1, activation='linear')  # Regression output
    ])
    model.compile(
        optimizer=keras.optimizers.Adam(learning_rate=lr),
        loss='mse',
        metrics=['mae']
    )
    return model

input_dim = X_train.shape[1]

# ==========================
# Hyperparameter Search
# ==========================
search_space = [
    {'units':64, 'lr':1e-3},
    {'units':128, 'lr':1e-3},
    {'units':64, 'lr':1e-4}
]

best_val_rmse = float('inf')
best_model = None
best_history = None
best_params = None
```

```
for params in search_space:
    print("\nTraining with parameters:", params)
    model = build_mlp(input_dim, units=params['units'], lr=params['lr'])
    history = model.fit(
        X_train, y_train,
        validation_data=(X_val, y_val),
        epochs=80,
        batch_size=32,
        verbose=1
    )

    val_preds = model.predict(X_val).flatten()
    val_rmse = math.sqrt(mean_squared_error(y_val, val_preds))
    print("Validation RMSE:", val_rmse)

    if val_rmse < best_val_rmse:
        best_val_rmse = val_rmse
        best_model = model
        best_history = history
        best_params = params

print("\nBest Parameters:", best_params)
print("Best Validation RMSE:", best_val_rmse)

# ==========================
# Test Set Evaluation
# ==========================
test_preds = best_model.predict(X_test).flatten()
test_rmse = math.sqrt(mean_squared_error(y_test, test_preds))
test_mae = mean_absolute_error(y_test, test_preds)
test_r2 = r2_score(y_test, test_preds)

print(f"\nTest Metrics:\nRMSE: {test_rmse:.4f}\nMAE: {test_mae:.4f}\nR²: {test_r2:.4f}")
```

```
# ==========================
# Save Model and Pipeline
# ==========================
model_path = "student_por_mlp_model.h5"
best_model.save(model_path)
pipeline_path = "student_por_preprocessor.joblib"
joblib.dump(preprocessor, pipeline_path)
print(f"\nSaved Model → {model_path}")
print(f"Saved Preprocessing Pipeline → {pipeline_path}")

# ==========================
# Training Curves & Prediction Plots
# ==========================
hist = best_history.history
# Loss Curve
plt.figure(figsize=(7,5))
plt.plot(hist['loss'], label='Train Loss')
plt.plot(hist['val_loss'], label='Val Loss')
plt.title("Training vs Validation Loss (MSE)")
plt.xlabel("Epoch")
plt.ylabel("Loss")
plt.legend()
plt.show()

# MAE Curve
plt.figure(figsize=(7,5))
plt.plot(hist['mae'], label='Train MAE')
plt.plot(hist['val_mae'], label='Val MAE')
plt.title("Training vs Validation MAE")
plt.xlabel("Epoch")
plt.ylabel("MAE")
plt.legend()
plt.show()
```

```
# Predicted vs Actual
plt.figure(figsize=(6,6))
plt.scatter(y_test, test_preds, alpha=0.7)
plt.plot([0,20],[0,20],'r--')
plt.title("Predicted vs Actual G3 (Test Set)")
plt.xlabel("Actual G3")
plt.ylabel("Predicted G3")
plt.show()

# Residual Analysis
residuals = y_test - test_preds
plt.figure(figsize=(7,5))
sns.histplot(residuals, bins=20, kde=True, color='orange')
plt.title("Residuals Distribution (Test Set)")
plt.xlabel("Residual")
plt.ylabel("Count")
plt.show()

plt.figure(figsize=(7,5))
plt.scatter(y_test, residuals, alpha=0.6)
plt.axhline(0, color='red', linestyle='--')
plt.title("Residuals vs Actual G3")
plt.xlabel("Actual G3")
plt.ylabel("Residuals")
plt.show()

# ==========================
# Metrics Summary
# ==========================
metrics_df = pd.DataFrame({
    "Metric": ["Test RMSE", "Test MAE", "Test R²"],
    "Value": [test_rmse, test_mae, test_r2]
})
display(metrics_df)
```