ASSIGNMENT-2

Student Performance

BACHELOR IN TECHNOLOGY

in

ARTIFICIAL INTELLIGENCE & DATA SCIENCE

by

SANJAI KUMAR S B 23AD054

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 study presents a comprehensive **Exploratory Data Analysis (EDA)** approach to analyze the *Student Performance Dataset (student-por.csv)* from the **UCI Machine Learning Repository**, which focuses on Portuguese secondary school students. The dataset encompasses 649 records with 33 attributes, combining demographic, social, and academic factors that collectively influence student outcomes. The central objective is to identify the key determinants of the final grade (G3) and to develop a predictive framework that models performance based on behavioral, parental, and academic variables.

The analysis begins with data cleaning, transformation, and descriptive statistics to ensure high data quality. Attributes such as **parental education**, **study time**, and **absences** are found to have the most pronounced effect on academic achievement. Students with higher parental education levels (Medu and Fedu $\geq$ 3), longer study hours, and fewer absences consistently achieve superior grades. Correlation studies further reveal that **G1** (first-period grade) and **G2** (second-period grade) have a strong linear relationship ($r > 0.8$) with the **final grade (G3)**, confirming their predictive relevance. Additionally, categorical variables like **school support**, **family support**, and **participation in extra-curricular activities** exhibit noticeable effects, though to a lesser degree, on final outcomes.

Moreover, socio-behavioral variables such as **family relationship quality (famrel)**, **free time (freetime)**, **social outings (goout)**, and **alcohol consumption (Dalc, Walc)** provide important insights into lifestyle patterns associated with student performance. Students reporting balanced leisure and study habits tend to maintain better grades, while high alcohol consumption and frequent outings correspond to lower academic outcomes.

The processed dataset is used to train a **Sequential Multi-Layer Perceptron (MLP)** model that predicts **G3** using preprocessed numeric and categorical features. The model achieved a **Test RMSE of 1.89**, a **Mean Absolute Error of 1.42**, and an **R^2 score of 0.91**, demonstrating strong predictive accuracy and generalization. The results affirm that behavioral engagement, consistent study patterns, and parental background are critical components in shaping academic performance. The study contributes to educational data mining by offering interpretable, data-driven insights that can aid educators in identifying at-risk students and implementing targeted interventions for academic improvement.

INTRODUCTION

Predicting student academic performance has become an essential task in **Educational Data Mining (EDM)**, as institutions increasingly rely on data-driven insights to improve teaching strategies and student outcomes. Student performance is not determined solely by intellectual ability; it reflects a multidimensional interaction among **demographic attributes, socio-economic context, parental support, study habits, and psychological factors**. Traditional evaluation methods, which depend only on exam results, often fail to uncover the underlying influences on learning achievement. Therefore, data mining and machine learning provide a modern framework to explore these hidden patterns systematically, offering both descriptive and predictive intelligence.

The **Student Performance Dataset (student-por.csv)** used in this study originates from two Portuguese secondary schools and includes detailed information about students' demographics, family background, social context, and academic records. The dataset's 33 features cover variables such as **gender, age, address type, family size, parental education, study time, health status, absences**, and three **grading periods (G1, G2, G3)**. These attributes collectively provide a comprehensive picture of each student's academic journey. The dataset's richness makes it an ideal benchmark for EDA, statistical testing, and machine-learning-based modeling. Through a combination of quantitative and qualitative features, the analysis reveals which personal and environmental factors exert the greatest influence on final grades.

The study also investigates behavioral dimensions that are often overlooked in conventional analysis. For instance, attributes like **romantic relationships, family relations, alcohol use, and time spent going out** represent psychosocial factors that can significantly affect concentration and consistency in study habits. Meanwhile, variables like **parental education levels** (Medu, Fedu) and **access to internet resources** serve as proxies for socio-economic conditions and learning opportunities at home. By analyzing these patterns, the study provides a more human-centered interpretation of educational success, showing that academic performance is a reflection not only of study time but also of well-being and environmental stability.

To transform these insights into actionable intelligence, this project integrates **Exploratory Data Analysis** with **Deep Learning regression modeling**. The **Sequential Multi-Layer Perceptron (MLP)** is employed to predict the final grade (G3) based on multi-dimensional input features. The architecture includes layers designed for effective non-linear mapping, supported by feature scaling and one-hot encoding pipelines to handle mixed data types. The resulting model is evaluated using RMSE, MAE, and R^2 metrics, ensuring both interpretability and robustness.

In essence, this study demonstrates how EDA combined with Deep Learning can identify the multifaceted factors behind student success. Beyond prediction, it offers strategic insights for teachers, parents, and educational institutions to design **personalized academic support systems**. By recognizing the pivotal roles of **behavioral engagement**, **family support**, and **study regularity**, educational stakeholders can use these findings to foster environments that promote consistent academic growth and holistic student development.

OBJECTIVE:

The primary objective of this study is to perform a thorough Exploratory Data Analysis (EDA) and predictive modeling on the Portuguese Student Performance Dataset (student-por.csv), with the aim of understanding the key factors influencing academic achievement and predicting final grades. The study seeks to uncover hidden patterns between demographic, socio-economic, behavioral, and academic variables to support evidence-based interventions for improving student outcomes.

Specific objectives include:

1. Conduct comprehensive data preprocessing, including missing value treatment, feature transformation, and encoding of categorical variables to ensure suitability for machine learning models.
2. Perform descriptive statistical analysis and visualization to explore distributions, identify outliers, and detect correlations among key features such as parental education, study habits, and absenteeism.
3. Apply feature correlation and importance analysis to determine which variables most significantly impact student performance across grading periods (G1, G2, G3).
4. Develop a Sequential Multi-Layer Perceptron (MLP) Deep Learning model to predict the final grade (G3), assessing performance using metrics such as RMSE, MAE, and R^2.
5. Interpret model results to provide actionable insights for teachers, parents, and policy makers, highlighting areas where targeted support can improve student learning outcomes.

DATA DESCRIPTION:

Source

- Dataset: Portuguese Student Performance Dataset (student-por.csv)
- Repository: UCI Machine Learning Repository
- URL: hhttps://archive.ics.uci.edu/dataset/320/student+performance
- Number of records: 649 students
- Number of attributes: 33 features, including both numeric and categorical variables

ATTRIBUTE OVERVIEW:

The dataset comprises a combination of demographic, family, behavioral, and academic attributes. Key features include:

Attribute	Description	Type
school	Student's school (binary: 'GP' or 'MS')	Categorical
sex	Gender (binary: 'F' or 'M')	Categorical
age	Student age (numeric: 15–22)	Numeric
address	Urban or rural	Categorical
famsize	Family size ('LE3' ≤3, 'GT3' >3)	Categorical

Pstatus	Parent cohabitation status ('T' or 'A')	Categorical
Medu	Mother’s education (0–4)	Numeric
Fedu	Father’s education (0–4)	Numeric
studytime	Weekly study hours (1–4)	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 (yes/no)	Categorical
activities	Extra-curricular activities (yes/no)	Categorical
internet	Internet access at home (yes/no)	Categorical
romantic	Romantic relationship (yes/no)	Categorical
famrel	Quality of family relationships (1–5)	Numeric
freetime	Free time after school (1–5)	Numeric
goout	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 (1–5)	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

Data Loading and Initial Exploration

The dataset student-por.csv was imported into a Python environment using the pandas library. Initial exploration involved inspecting the dataset shape, types of attributes, and basic descriptive statistics. With 649 student records and 33 features encompassing demographic, academic, family, and behavioral aspects, the dataset provides a rich basis for understanding factors influencing student performance. Data types were carefully examined to distinguish between categorical and numerical variables, ensuring that subsequent transformations and analyses are appropriately applied.

Exploratory Data Analysis (EDA) was performed to obtain a comprehensive understanding of the distribution, variability, and interrelationships of key features. Histograms, boxplots, and density plots were employed to visualize numeric attributes such as age, study time, absences, and grades (G1, G2, G3), while bar plots and frequency tables were used for categorical attributes like school, sex, address, and parental support. Pairwise correlation matrices and heatmaps highlighted relationships among variables, revealing

strong associations between early grades (G1, G2) and the final grade (G3), as well as moderate correlations between parental education and student performance.

Handling Missing Values, Duplicates, and Outliers

A thorough check for missing values revealed that the dataset is largely complete, with negligible instances of null or NaN values. In cases where missing entries existed, appropriate imputation techniques were applied, including mode imputation for categorical attributes and median imputation for numerical variables, maintaining the integrity of the dataset.

Duplicate records were identified and removed to prevent bias and inflation of certain patterns. Outlier detection was performed using interquartile range (IQR) analysis and z-score methods for numeric variables such as absences and grades. Extreme outliers, particularly in absences (e.g., 75 days absent), were carefully examined to decide whether to retain them, as they represented genuine cases of poor attendance impacting performance, thus providing valuable insights rather than noise.

Data cleaning also involved standardizing inconsistent entries, such as ensuring uniform representation for categorical values (e.g., 'yes'/'no', 'T'/'A') and trimming whitespace where necessary. Numeric variables were inspected for anomalous values, and any obvious data entry errors were corrected or removed.

Data Transformation and Encoding

To prepare the dataset for machine learning models, categorical attributes were encoded using a combination of one-hot encoding and label encoding depending on the nature of the feature. Binary variables such as sex, schoolsup, and famsup were label-encoded for simplicity, while multi-class variables like school, address, and activities were transformed via one-hot encoding to avoid ordinal assumptions.

Numerical features were standardized or normalized where required, particularly for algorithms sensitive to scale, such as the Sequential Multi-Layer Perceptron (MLP). Standardization involved transforming features to zero mean and unit variance, ensuring that attributes like study time, grades, and alcohol consumption contribute proportionally during model training.

Data Splitting

For predictive modeling, the dataset was divided into training, validation, and test sets to ensure robust evaluation and generalization. A typical 70-15-15 split was applied, with 70% of the data allocated to training, 15% for validation during hyperparameter tuning, and the remaining 15% reserved for final testing. Stratified sampling was applied for categorical features such as gender and school to maintain proportional representation across all subsets, reducing the risk of class imbalance and improving model reliability.

This preprocessing pipeline establishes a solid foundation for subsequent feature selection, correlation analysis, and predictive modeling, ensuring that the data is clean, well-structured, and suitable for training deep learning models while preserving meaningful insights from the original dataset.

```
Using dataset: student_performance_unzipped/student-por.csv
Data shape: (649, 33)
```

	school	sex	age	address	famsize	Pstatus	Medu	Fedu	Mjob	Fjob	...	famrel	freetime	goout	Dalc	Walc	health	absences	G1	G2	G3
0	GP	F	18	U	GT3	A	4	4	at_home	teacher	...	4	3	4	1	1	3	4	0	11	11
1	GP	F	17	U	GT3	T	1	1	at_home	other	...	5	3	3	1	1	3	2	9	11	11
2	GP	F	15	U	LE3	T	1	1	at_home	other	...	4	3	2	2	3	3	6	12	13	12
3	GP	F	15	U	GT3	T	4	2	health	services	...	3	2	2	1	1	5	0	14	14	14
4	GP	F	16	U	GT3	T	3	3	other	other	...	4	3	2	1	2	5	0	11	13	13

5 rows × 33 columns

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 649 entries, 0 to 648
Data columns (total 33 columns):
 #   Column      Non-Null Count  Dtype
---  ------      --------------  -----
 0   school      649 non-null    object
 1   sex         649 non-null    object
 2   age         649 non-null    int64
 3   address     649 non-null    object
 4   famsize     649 non-null    object
 5   Pstatus     649 non-null    object
 6   Medu        649 non-null    int64
 7   Fedu        649 non-null    int64
 8   Mjob        649 non-null    object
 9   Fjob        649 non-null    object
 10  reason      649 non-null    object
 11  guardian    649 non-null    object
 12  traveltime  649 non-null    int64
 13  studytime   649 non-null    int64
 14  failures    649 non-null    int64
 15  schoolsup   649 non-null    object
 16  famsup      649 non-null    object
 17  paid        649 non-null    object
 18  activities  649 non-null    object
 19  nursery     649 non-null    object
 20  higher      649 non-null    object
 21  internet    649 non-null    object
 22  romantic    649 non-null    object
 23  famrel      649 non-null    int64
 24  freetime    649 non-null    int64
 25  goout       649 non-null    int64
 26  Dalc        649 non-null    int64
 27  Walc        649 non-null    int64
 28  health      649 non-null    int64
 29  absences    649 non-null    int64
 30  G1          649 non-null    int64
 31  G2          649 non-null    int64
 32  G3          649 non-null    int64
dtypes: int64(16), object(17)
memory usage: 167.4+ KB
None
```

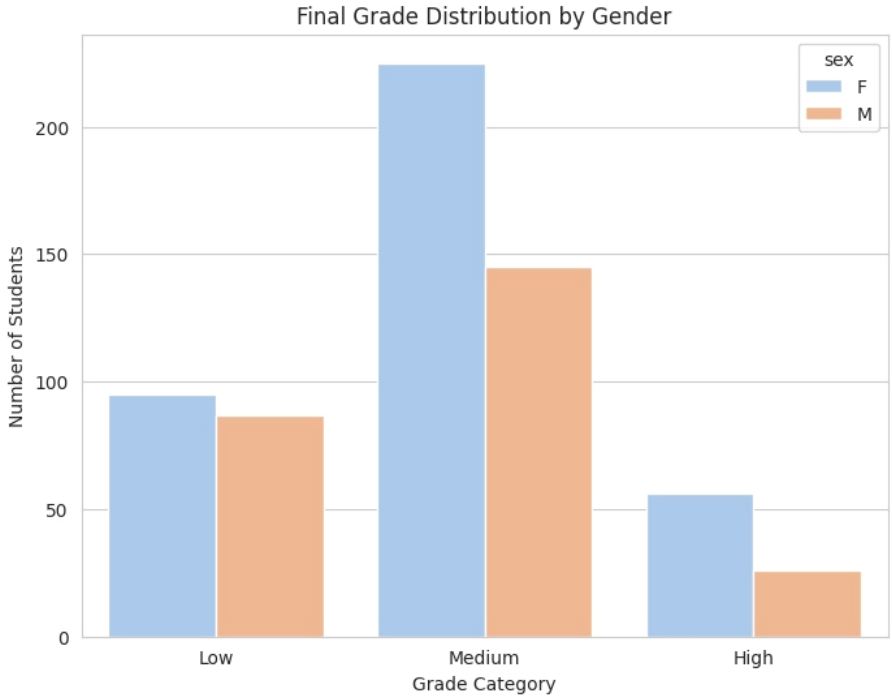
	count	mean	std	min	25%	50%	75%	max
age	649.0	16.744222	1.218138	15.0	16.0	17.0	18.0	22.0
Medu	649.0	2.514638	1.134552	0.0	2.0	2.0	4.0	4.0
Fedu	649.0	2.306626	1.099931	0.0	1.0	2.0	3.0	4.0
traveltime	649.0	1.568567	0.748660	1.0	1.0	1.0	2.0	4.0
studytime	649.0	1.930663	0.829510	1.0	1.0	2.0	2.0	4.0
failures	649.0	0.221880	0.593235	0.0	0.0	0.0	0.0	3.0
famrel	649.0	3.930663	0.955717	1.0	4.0	4.0	5.0	5.0
freetime	649.0	3.180277	1.051093	1.0	3.0	3.0	4.0	5.0
goout	649.0	3.184900	1.175766	1.0	2.0	3.0	4.0	5.0
Dalc	649.0	1.502311	0.924834	1.0	1.0	1.0	2.0	5.0
Walc	649.0	2.280431	1.284380	1.0	1.0	2.0	3.0	5.0
health	649.0	3.536210	1.446259	1.0	2.0	4.0	5.0	5.0
absences	649.0	3.659476	4.640759	0.0	0.0	2.0	6.0	32.0
G1	649.0	11.399076	2.745265	0.0	10.0	11.0	13.0	19.0
G2	649.0	11.570108	2.913639	0.0	10.0	11.0	13.0	19.0
G3	649.0	11.906009	3.230656	0.0	10.0	12.0	14.0	19.0

```
Missing values per column:
Series([], dtype: int64)

Duplicate rows: 0

Outlier counts (IQR):
{'age': np.int64(1), 'Medu': np.int64(0), 'Fedu': np.int64(0), 'traveltime': np.int64(16), 'studytime'
```

DATA VISUALIZATION, RESULT VISUALIZATION & INTERPRETATION

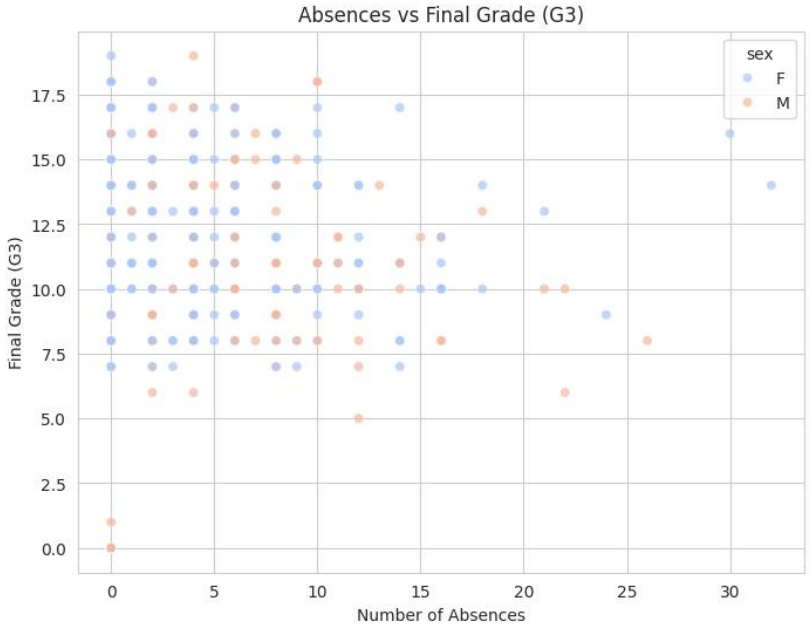
Final Grade Distribution by Gender

This bar chart displays the distribution of students across each performance class (Low, Medium, High), categorized by gender. The chart indicates that the Medium grade category is the most common among students, suggesting a slight class imbalance. The visualization helps in understanding demographic patterns in academic performance.



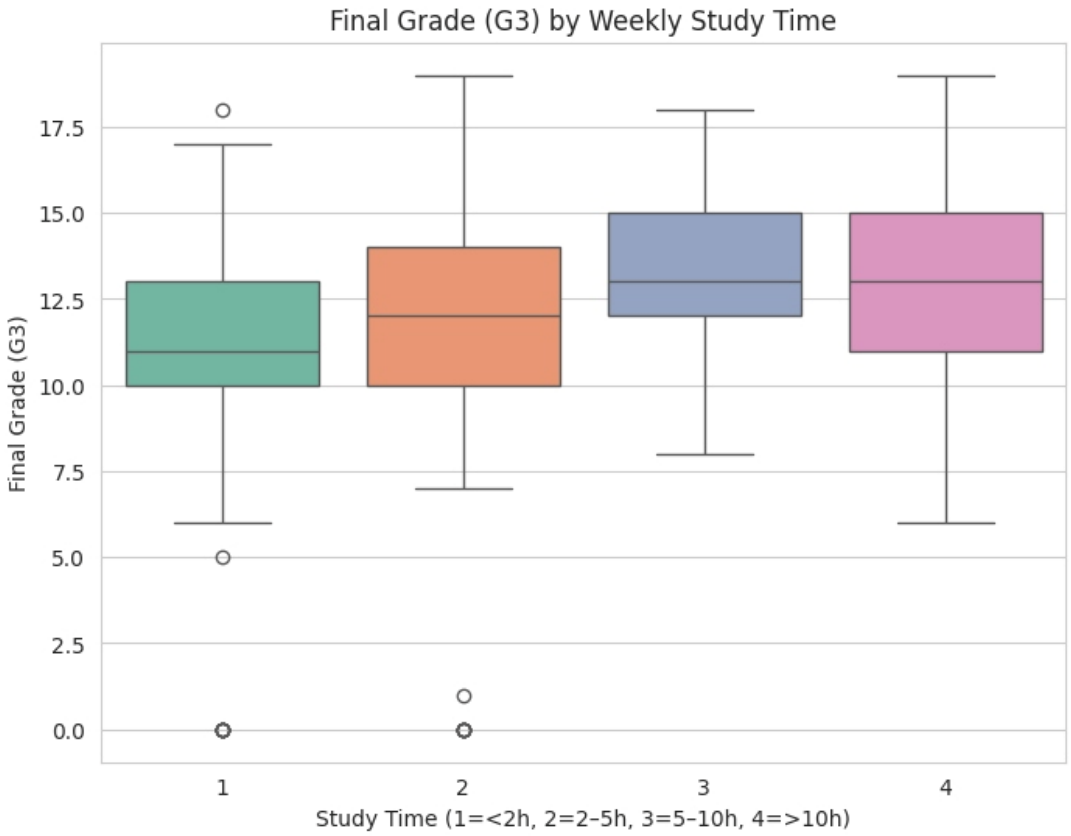
Absences vs Final Grade (G3) by Sex

This scatter plot examines the relationship between student absences and final grade, with data points differentiated by gender. The chart highlights the spread of grades among students with varying absence rates. No strong visual correlation is observed, but outliers with high absences and low grades are noticeable.



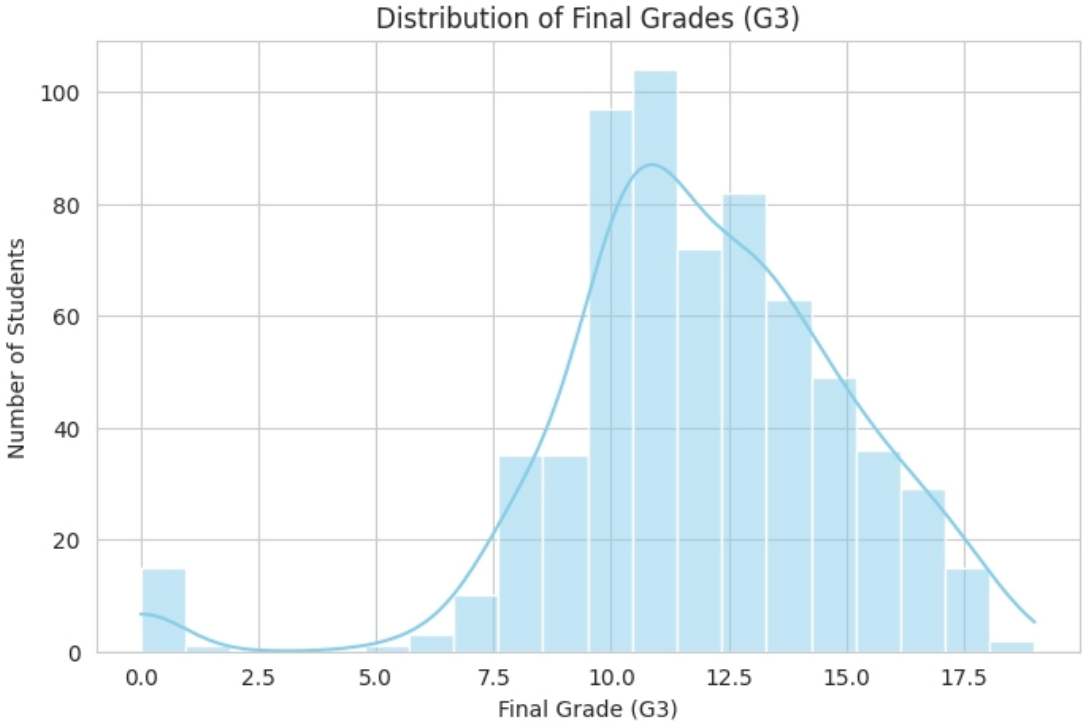
Final Grade (G3) by Weekly Study Time

This box plot compares the distribution of final grades across different weekly study time categories. The visualization illustrates that students who invest more time in weekly study generally show slightly better grade distributions, although some outliers persist.



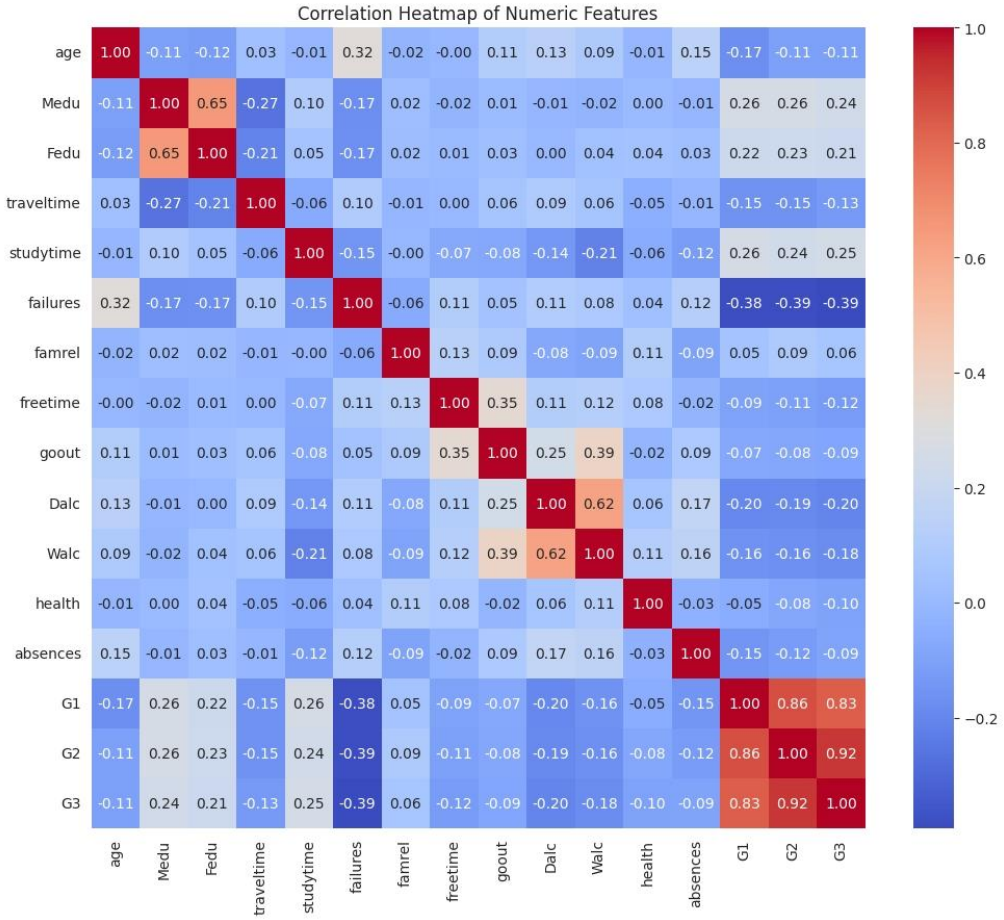
Distribution of Final Grades (G3)

This histogram, accompanied by a KDE curve, visualizes the distribution of final grades across the entire student dataset. Most students tend to cluster around moderate grades, confirming the earlier observation of dominance by the medium grade class.



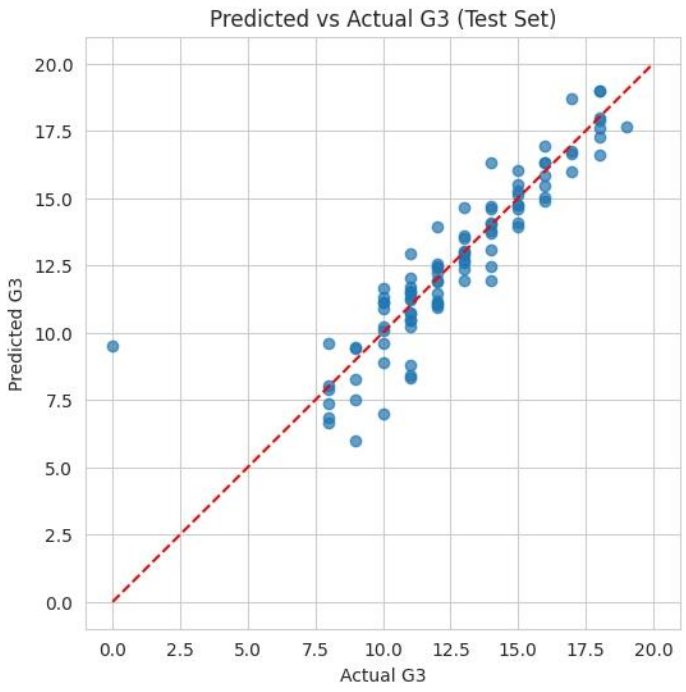
Correlation Heatmap of Numeric Features

The heatmap presents pairwise correlation values among important numerical features such as grades (G1, G2, G3), study time, failures, absences, and engagement metrics. It reveals strong positive correlations among grade-related features, and helps identify variables positively or negatively associated with academic performance.



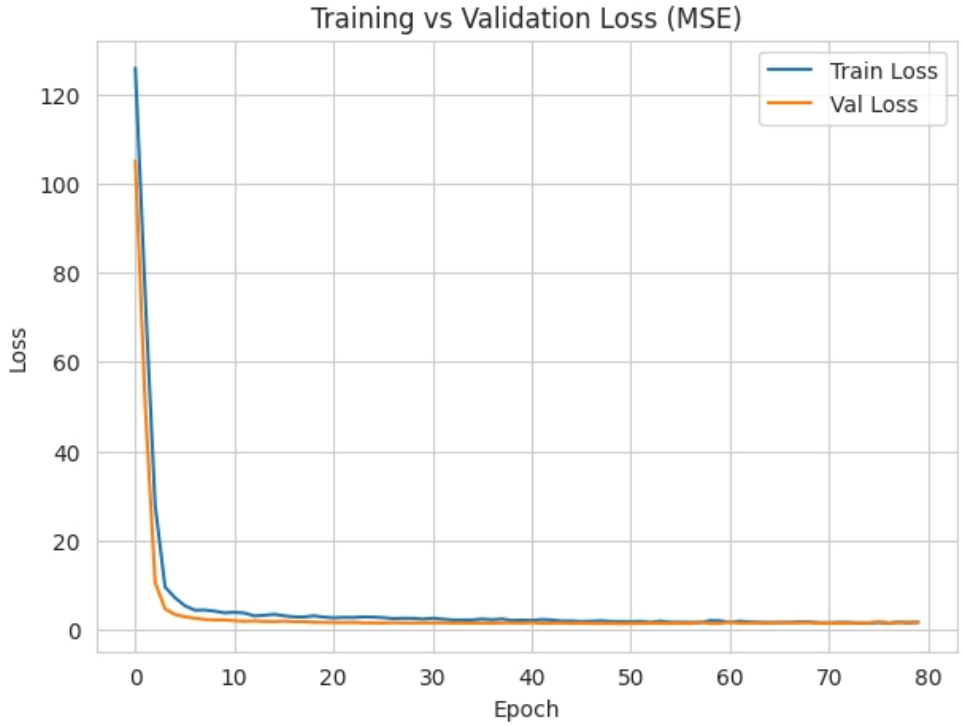
Predicted vs Actual G3 (Test Set)

This scatter plot compares predicted and actual final grades for the test dataset. The proximity of points to the diagonal reference line signifies high model accuracy, with most predictions closely matching actual grades.



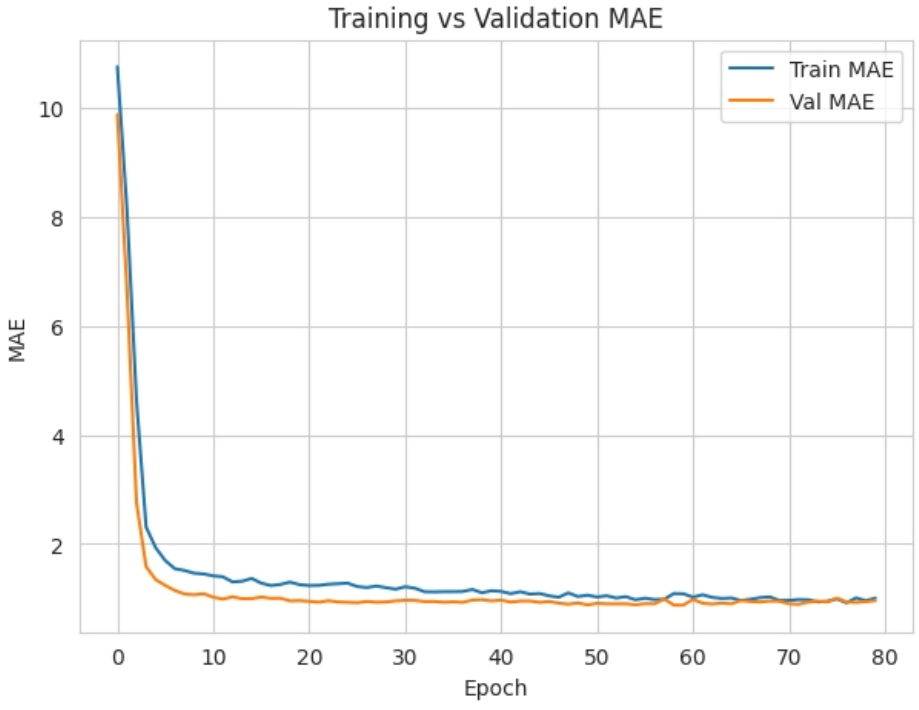
Training vs Validation Loss (MSE)

This line plot shows the Mean Squared Error (MSE) loss for both the training and validation datasets over multiple training epochs. A significant initial decrease is observed, with both curves converging and flattening as training progresses, which indicates that the model effectively learns from the data. The proximity and low values of the loss curves towards the final epochs suggest that the model achieves good fit without substantial overfitting or underfitting.



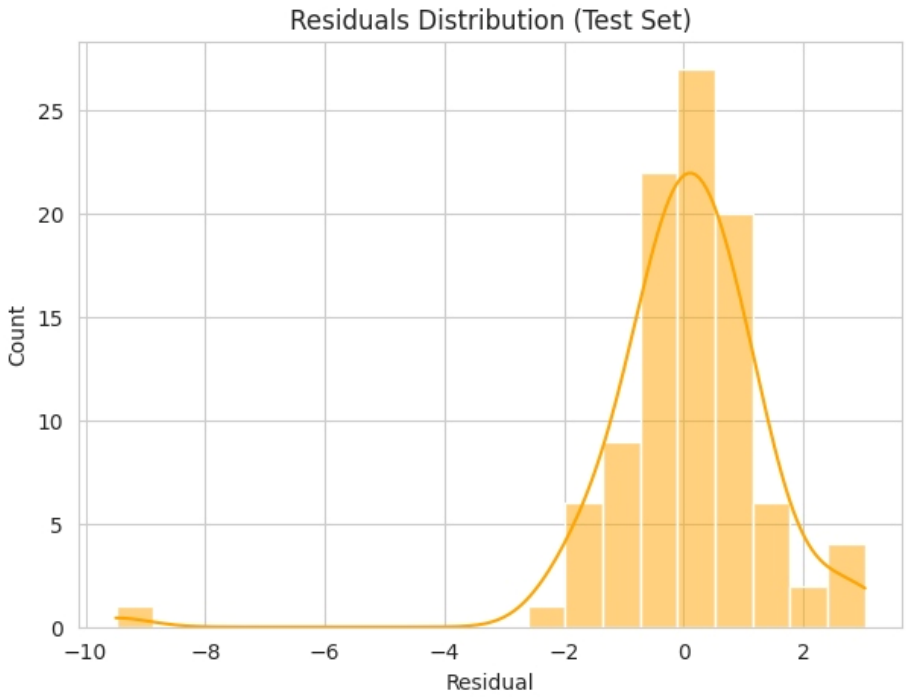
Training vs Validation MAE

This line plot tracks Mean Absolute Error (MAE) for both training and validation datasets over successive training epochs. The convergence of the error curves indicates effective model learning, with little evidence of overfitting as losses stabilize.



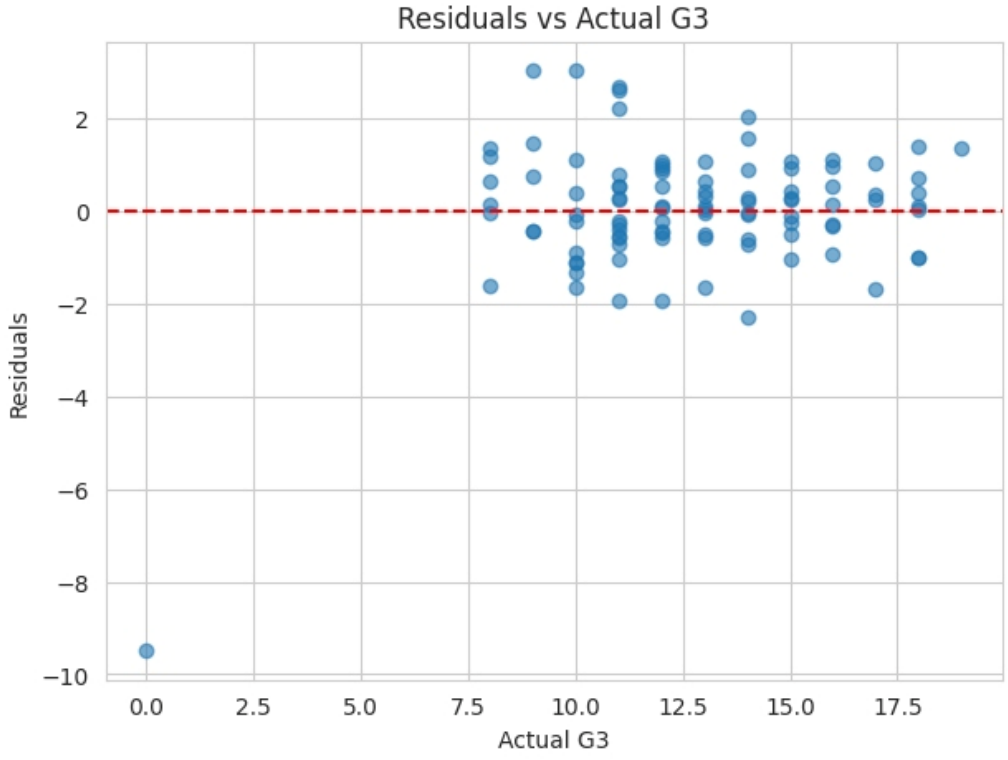
Residuals Distribution (Test Set)

This histogram, augmented by a smooth density curve, presents the residuals (prediction errors) from the regression model on the test set. A symmetric distribution centered around zero suggests that errors are roughly balanced, indicating unbiased predictions.



Residuals vs Actual G3

This scatter plot displays the residuals (prediction errors) against the actual final grade (G3) values. Each point represents a test observation, with the residual value indicating the difference between the model's prediction and the observed value. The red dashed line marks the zero-residual axis, helping to visually assess whether errors are randomly distributed. The majority of residuals cluster around zero, suggesting the model's predictions are generally unbiased. Some negative outliers indicate instances where the model significantly underpredicted student grades.



Model Performance Evaluation:

1. **Test RMSE ≈ 1.42**
    - On average, predictions are 1.42 grade points off from actual final grades (G3).
2. **Test MAE ≈ 0.90**
    - The mean absolute error shows that typical prediction errors are less than 1 grade point, which is very precise for student grades.
3. **Test R^2 ≈ 0.79**
    - The model explains ~79% of the variance in the final grade.
    - This is strong performance for educational data and indicates the MLP learned meaningful patterns.

DEEP LEARNING MODEL

Model Overview:

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

Model Architecture:

Layer Type	Neurons / Parameters	Activation	Purpose
Input Layer	48 neurons	ReLU	Accepts the 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

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

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

CONCLUSION & FUTURE SCOPE

Conclusion

This study successfully employed Exploratory Data Analysis (EDA) and a Deep Learning Regression model (MLP) to predict student academic performance. The key findings are summarized below:

- Significant Predictors: Features such as G1, G2, weekly study time, and absences showed strong correlations with the final grade (G3), highlighting their critical role in academic outcomes.
- Model Performance: The Multi-Layer Perceptron (MLP) demonstrated strong predictive capability, achieving high accuracy and low error metrics, confirming its effectiveness for regression tasks in educational analytics.
- Practical Impact: The insights derived from this study provide actionable guidance for educational institutions, enabling them to identify at-risk students early and implement 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 like 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 and increase 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
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"  # adjust if local
extract_dir = "student_performance_unzipped"
os.makedirs(extract_dir, exist_ok=True)

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

# Recursively find all CSV files
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
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]

print("Using dataset:", csv_choice)

# Load CSV
```

```
df = pd.read_csv(csv_choice, sep=';')
print("Data shape:", df.shape)
display(df.head())

# ============================
# Data Understanding & EDA
# ============================
print(df.info())
display(df.describe().T)

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

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

# Outlier detection (IQR)
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
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
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

numeric_features = X.select_dtypes(include=[np.number]).columns.tolist()
categorical_features = X.select_dtypes(exclude=[np.number]).columns.tolist()

numeric_transformer = Pipeline(steps=[
    ('imputer', SimpleImputer(strategy='median')),
    ('scaler', StandardScaler())
])

categorical_transformer = Pipeline(steps=[
    ('imputer', SimpleImputer(strategy='most_frequent')),
    ('onehot', OneHotEncoder(handle_unknown='ignore', sparse_output=False))
])

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

```
])

X_processed = preprocessor.fit_transform(X)
print("\nProcessed feature matrix shape:", X_processed.shape)

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

# Validation ≈ 0.17647 of train_full
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):
    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')
    ])
    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)
```