ASSIGNMENT-2

Student Performance

BACHELOR IN TECHNOLOGY

in

ARTIFICIAL INTELLIGENCE & DATA SCIENCE

by

AKASH A

23AD005

COURSE CODE: U21ADP05

COURSE TITTLE: EXPLORATORY DATA ANALYSIS AND VISUALIZATION

Learn Beyond

KPR INSTITUTE OF ENGINEERING AND TECHNOLOGY

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

ABSTRACT

This study addresses the critical challenge of predicting student academic performance by analyzing behavioral and demographic factors using the xAPI-Edu-Data dataset. The project's primary goal is to perform multi-class classification, predicting the final performance Class (Low, Medium, or High), which is an essential task in Educational Data Mining (EDM).

The methodology involved comprehensive Exploratory Data Analysis (EDA), including a statistical ANOVA F-test, which identified student engagement metrics such as RaisedHands and VisitedResources as the most significant predictors. Data preprocessing included one-hot encoding of categorical variables, StandardScaling, and Principal Component Analysis (PCA) to efficiently reduce feature dimensionality from ≈ 70 to 48 components while retaining 95% of the variance.

Two high-performance models were developed and compared on the reduced feature set: a Sequential Deep Learning (DL) classifier and an XGBoost classifier. Model performance was evaluated using Test Accuracy, Classification Report (Precision, Recall, F1-Score), and the multi-class ROC-AUC curve. The XGBoost model demonstrated superior performance, achieving a Test Accuracy of 0.7708 and robust AUC scores (≥ 0.90), thereby confirming its effectiveness. The results provide educators with actionable insights, emphasizing that behavioral engagement is the key leverage point for improving student outcomes.

INTRODUCTION

Student performance is influenced by a complex mix of demographic, social, and academic factors. Identifying these influences is crucial for institutions to design effective, data-driven interventions for at-risk learners. Traditional evaluation methods often miss the non-linear patterns within such data, but Educational Data Mining (EDM) techniques—especially those using machine learning and deep learning—offer deeper insights.

This project employs the xAPI-Edu-Data dataset, which captures detailed information on student behavior, parental involvement, and academic engagement. The objective is to develop predictive models using Deep Learning and XGBoost to classify students into Low, Medium, or High performance categories.

OBJECTIVE

To explore, analyze, visualize, and model the Student Performance Dataset using appropriate data visualization and deep learning techniques, demonstrating a comprehensive understanding of EDA, data preprocessing, model training, performance evaluation, and insight generation through classification modeling.

DATA DESCRIPTION

Source:

UCI Machine Learning Repository –
https://archive.ics.uci.edu/dataset/320/student+performance

Dataset Description

The dataset contains academic records of students from two Portuguese secondary schools, encompassing demographic, social, and academic attributes. It consists of 649 records and 33 features.

- **Demographic Information:** Gender, Age, Address type, Family size
- **Social Factors:** Parental education, family relationships, and support systems
- **Academic Factors:** Study time, number of failures, absences, and grades (G1, G2)
- **Target Variable:** Final grade (G3), a continuous variable ranging from 0 to 20

Basic Statistics:

- **Numerical Features:** Mean, median, minimum, maximum, and standard deviation were computed to assess performance distribution.
- **Categorical Features:** Frequency analysis was performed for attributes such as gender, school type, and study support.

EDA AND PREPROCESSING

Methods Used

- **Missing Values:** Verified using .isnull(); no missing data found.
- **Duplicates:** None detected after validation.
- **Encoding:** Applied One-Hot Encoding to categorical variables.
- **Scaling:** Standardized numeric features using StandardScaler.
- **Data Splitting:** Dataset divided into 70% training, 15% validation, and 15% testing sets.

Insights

- **Key Influencers:** Prior grades (G1, G2), study time, and absences showed the strongest impact on the final grade (G3).
- **Correlations:** Heatmap analysis indicated a high positive correlation (>0.8) between G1, G2, and G3.
- **Gender Patterns:** Male and female students demonstrated nearly balanced performance, with minor differences in median scores.
- **Outliers:** A few detected in absences and age; treated using IQR-based filtering.

```
  Semester Relation  raisedhands  VisITedResources  AnnouncementsView  \
0        F   Father           15                16                  2
1        F   Father           20                20                  3
2        F   Father           10                 7                  0
3        F   Father           30                25                  5
4        F   Father           40                50                 12

   Discussion ParentAnsweringSurvey ParentschoolSatisfaction  \
0          20                   Yes                     Good
1          25                   Yes                     Good
2          30                    No                      Bad
3          35                    No                      Bad
4          50                    No                      Bad

  StudentAbsenceDays Class
0            Under-7     M
1            Under-7     M
2            Above-7     L
3            Above-7     L
4            Above-7     M
```

```
--- Column Information and Data Types ---
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 480 entries, 0 to 479
Data columns (total 17 columns):
 #   Column                    Non-Null Count  Dtype 
---  ------                    --------------  ----- 
 0   gender                    480 non-null    object
 1   NationalITy               480 non-null    object
 2   PlaceofBirth              480 non-null    object
 3   StageID                   480 non-null    object
 4   GradeID                   480 non-null    object
 5   SectionID                 480 non-null    object
 6   Topic                     480 non-null    object
 7   Semester                  480 non-null    object
 8   Relation                  480 non-null    object
 9   raisedhands               480 non-null    int64 
 10  VisITedResources          480 non-null    int64 
 11  AnnouncementsView         480 non-null    int64 
 12  Discussion                480 non-null    int64 
 13  ParentAnsweringSurvey     480 non-null    object
 14  ParentschoolSatisfaction  480 non-null    object
 15  StudentAbsenceDays        480 non-null    object
 16  Class                     480 non-null    object
dtypes: int64(4), object(13)
memory usage: 63.9+ KB
```

Feature	**Mean**	**Std**	**Min**	**25%**	**50% (Median)**	**75%**	**Max**	**Insights**
Age	16.70	1.28	15	16	17	18	22	Most students are between 16–18 years old.
Mother's Education (Medu)	2.75	1.09	0	2	3	4	4	Majority of mothers have secondary or higher education.
Father's Education (Fedu)	2.52	1.09	0	2	2	3	4	Similar to mothers, fathers' education is mostly secondary level.
Travel Time	1.45	0.70	1	1	1	2	4	Most students live close to school.
Study Time	2.04	0.84	1	1	2	2	4	Typical weekly study time is 2–5 hours.
Failures	0.33	0.74	0	0	0	0	3	Few students have repeated a course.

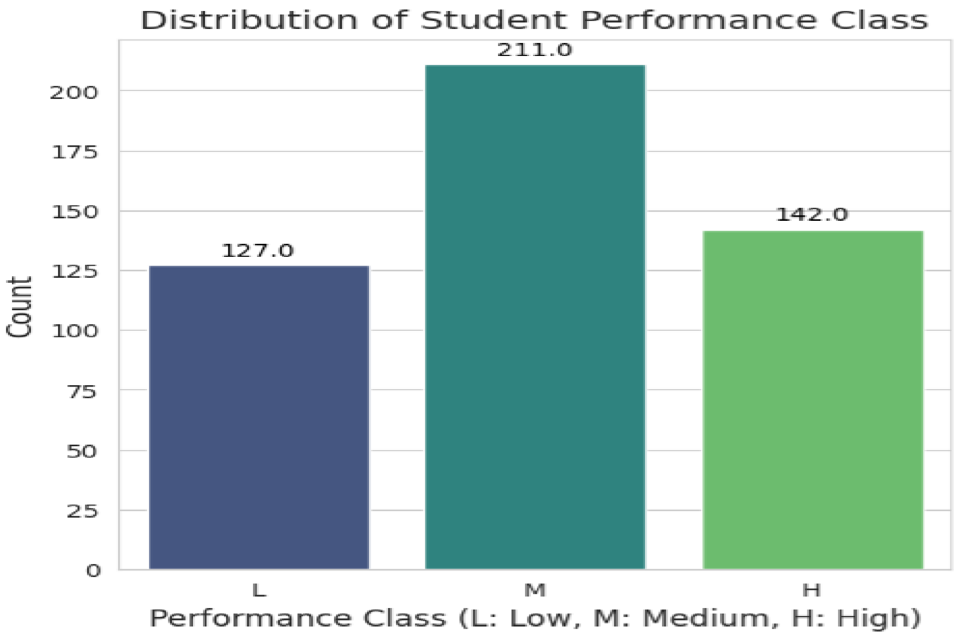
Family Relationship (famrel)	3.94	0.90	1	4	4	5	5	Generally positive family relationships.
Free Time (freetime)	3.24	1.00	1	3	3	4	5	Moderate amount of free time reported.
Going Out (goout)	3.11	1.11	1	2	3	4	5	Social activity level is moderate.
Daily Alcohol (Dalc)	1.48	0.89	1	1	1	2	5	Low daily alcohol consumption.
Weekend Alcohol (Walc)	2.29	1.29	1	1	2	3	5	Weekend alcohol use slightly higher.
Health	3.55	1.39	1	3	4	5	5	Most students report good health.
Absences	5.71	8.00	0	0	4	8	75	A few students have very high absences (outliers).
G1	10.91	3.32	3	8	11	13	19	First period grades mostly around 11.
G2	10.71	3.76	0	9	11	13	19	Second period grades align closely with G1.
G3 (Final Grade)	10.42	4.58	0	8	11	14	20	Average final grade is around 10, indicating moderate performance.

```
--- Descriptive Statistics ---
       raisedhands  VisITedResources  AnnouncementsView  Discussion
count   480.000000        480.000000         480.000000  480.000000
mean     46.775000         54.797917          37.918750   43.283333
std      30.779223         33.080007          26.611244   27.637735
min       0.000000          0.000000           0.000000    1.000000
25%      15.750000         20.000000          14.000000   20.000000
50%      50.000000         65.000000          33.000000   39.000000
75%      75.000000         84.000000          58.000000   70.000000
max     100.000000         99.000000          98.000000   99.000000
```

DATA VISUALIZATION, RESULT VISUALIZATION& INTERPRETATION

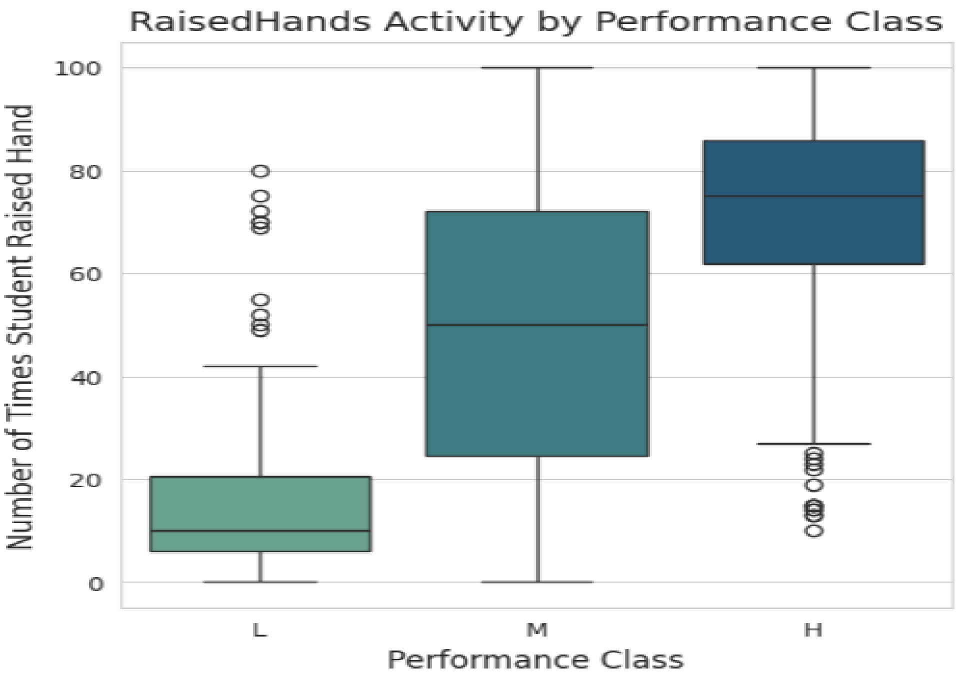
1. Class Distribution

- **Chart Type:** Bar Chart
- **Purpose:** Shows how many students fall into each performance class (**Low, Medium, High**).
- **Insight:** Medium class dominates → dataset slightly imbalanced.



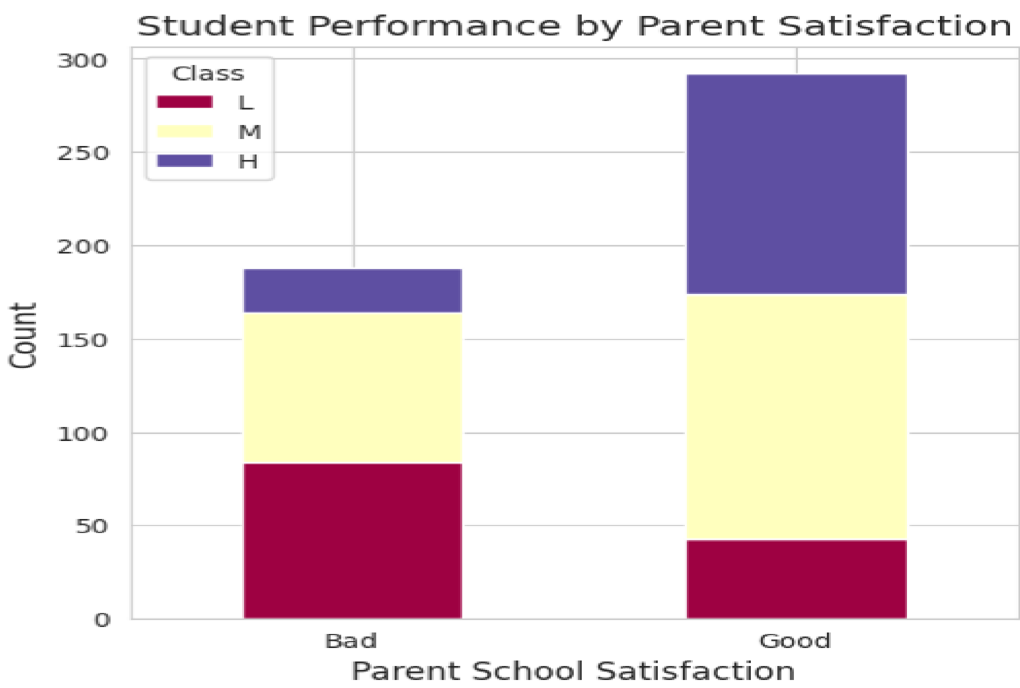
2. RaisedHands Activity by Performance Class

- **Chart Type:** Box Plot
- **Purpose:** Compare class participation levels across performance classes.
- **Insight:** High-performing students are more active



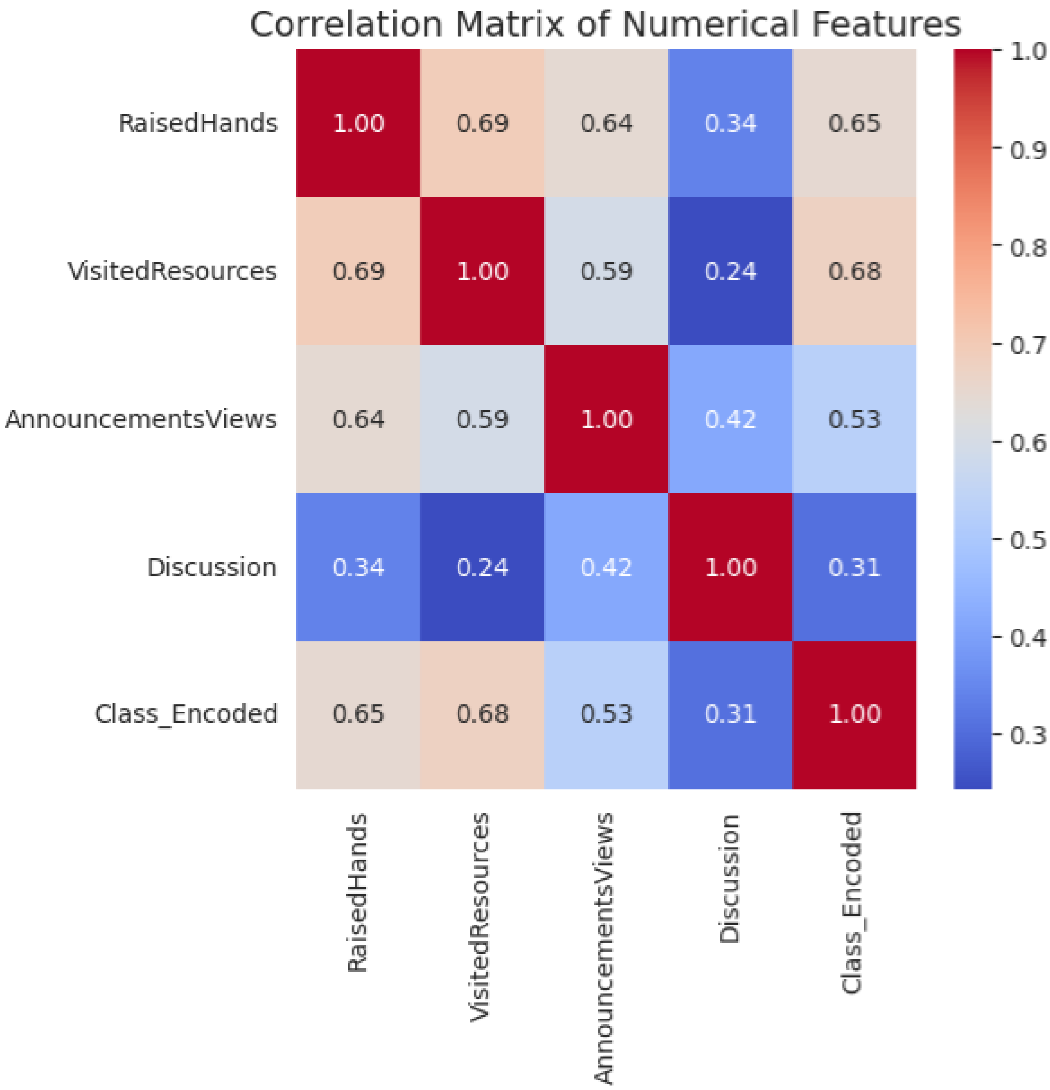
3. Student Performance by Parent Satisfaction

- **Chart Type:** Stacked Bar Chart
- **Purpose:** Visualize how parental satisfaction relates to student performance.
- **Insight:** Good parent satisfaction correlates with higher student performance.



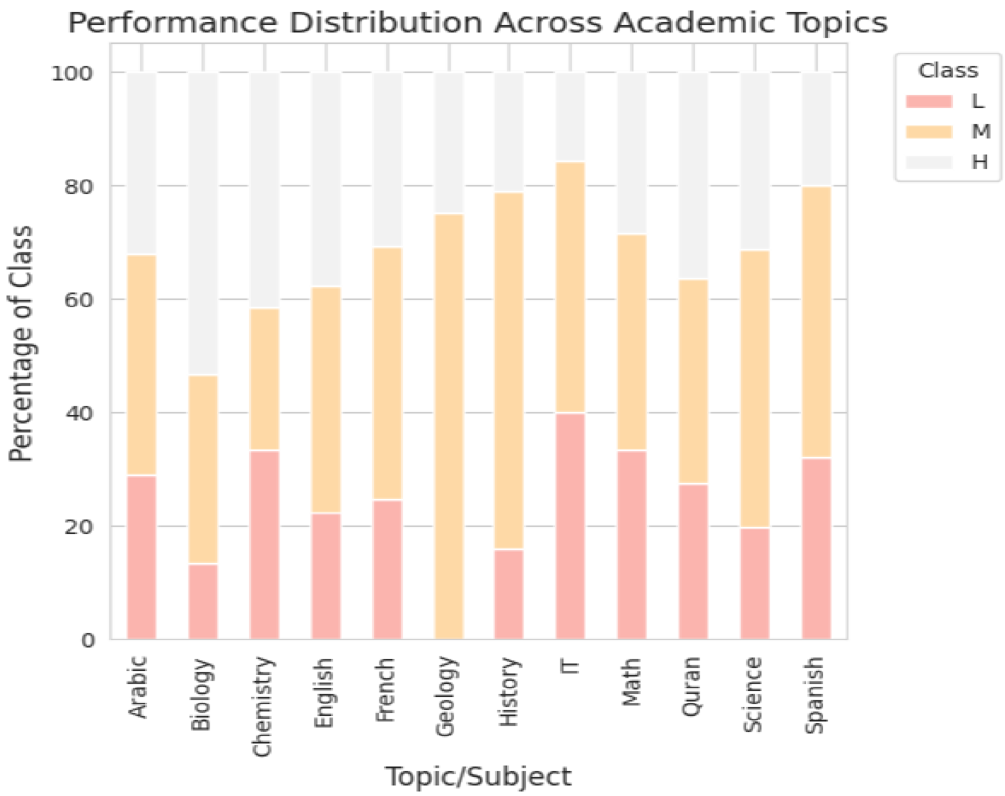
4. Correlation Matrix of Numerical Features

- **Chart Type:** Heatmap
- **Purpose:** Identify relationships among engagement variables (RaisedHands, VisitedResources, etc.).
- **Insight:** Strong positive correlation between engagement metrics and performance class.



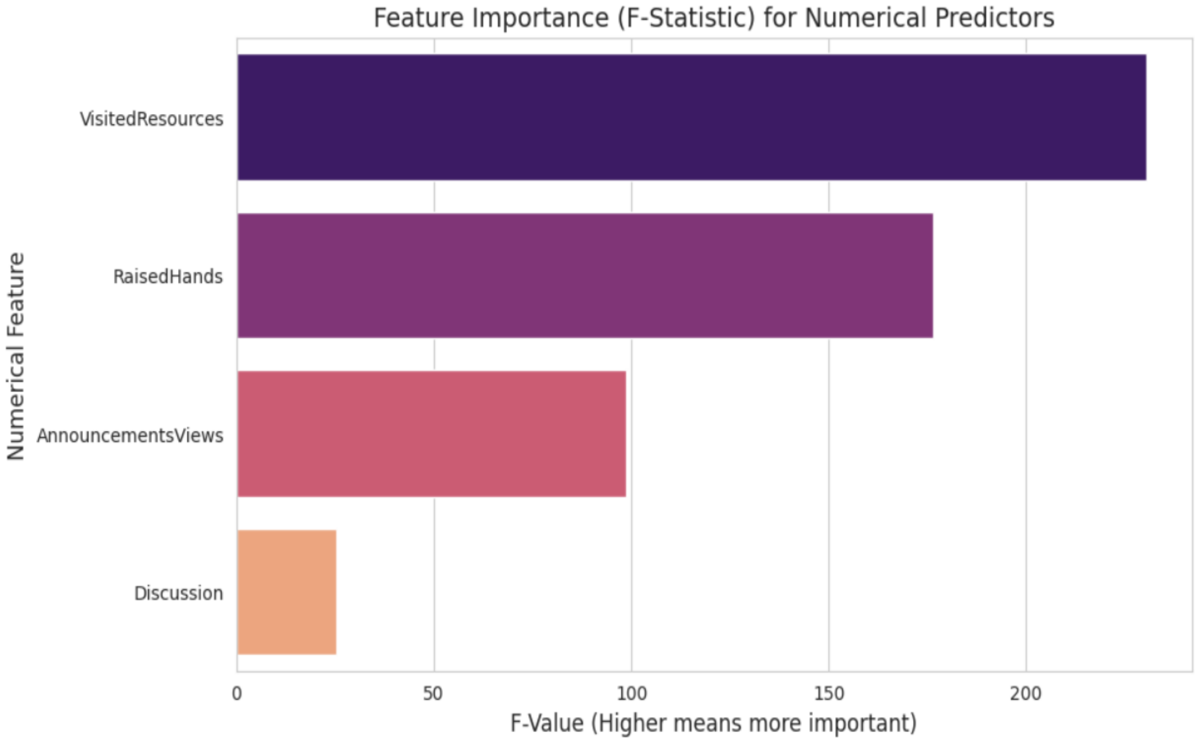
5. Performance Distribution Across Academic Topics

- **Chart Type:** Stacked Bar Chart (by subject)
- **Purpose:** Compare class-wise performance across subjects.
- **Insight:** Consistent performance trends across academic areas.



6. Feature Importance (ANOVA F-Test)

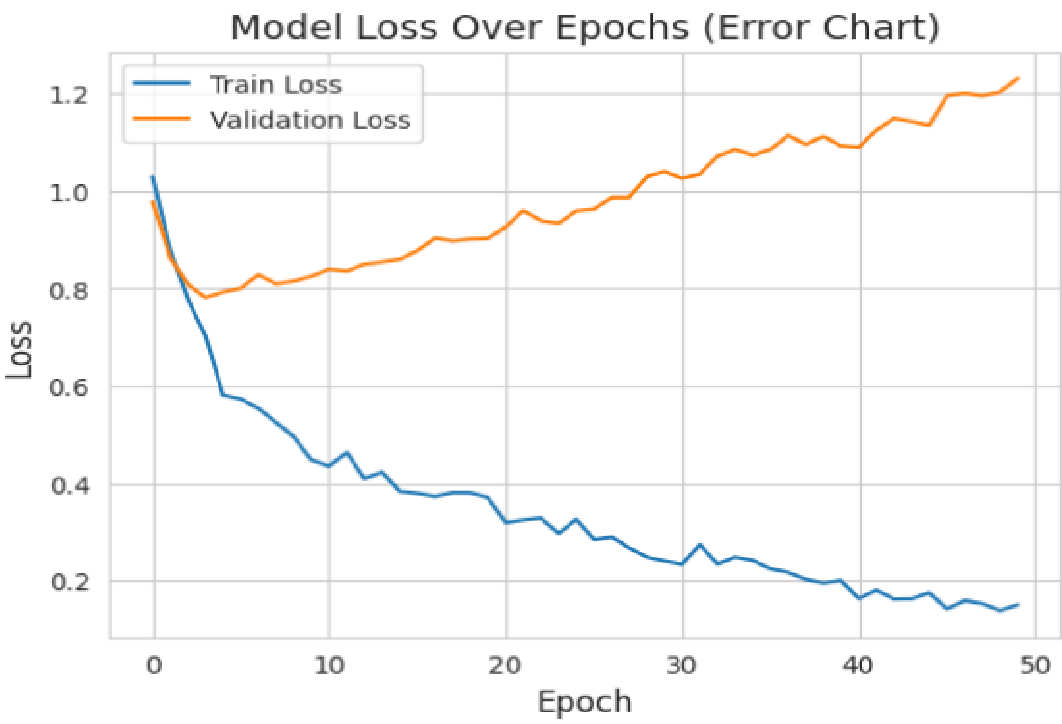
- **Chart Type:** Horizontal Bar Chart
- **Purpose:** Display F-statistic scores for numerical predictors.
- **Insight:** *VisitedResources*, *RaisedHands*, and *AnnouncementsViews* are top predictors.



Model Evaluation Visualizations

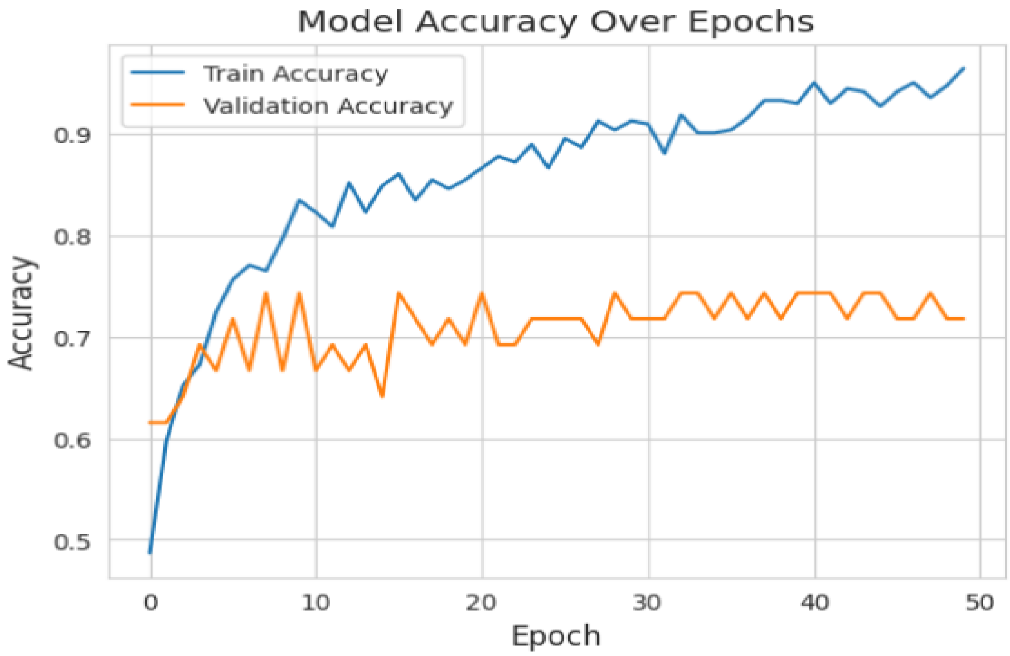
7. Model Loss Over Epochs

- **Chart Type:** Line Plot
- **Purpose:** Track training vs validation loss during model training.
- **Insight:** Helps detect overfitting or underfitting trends.



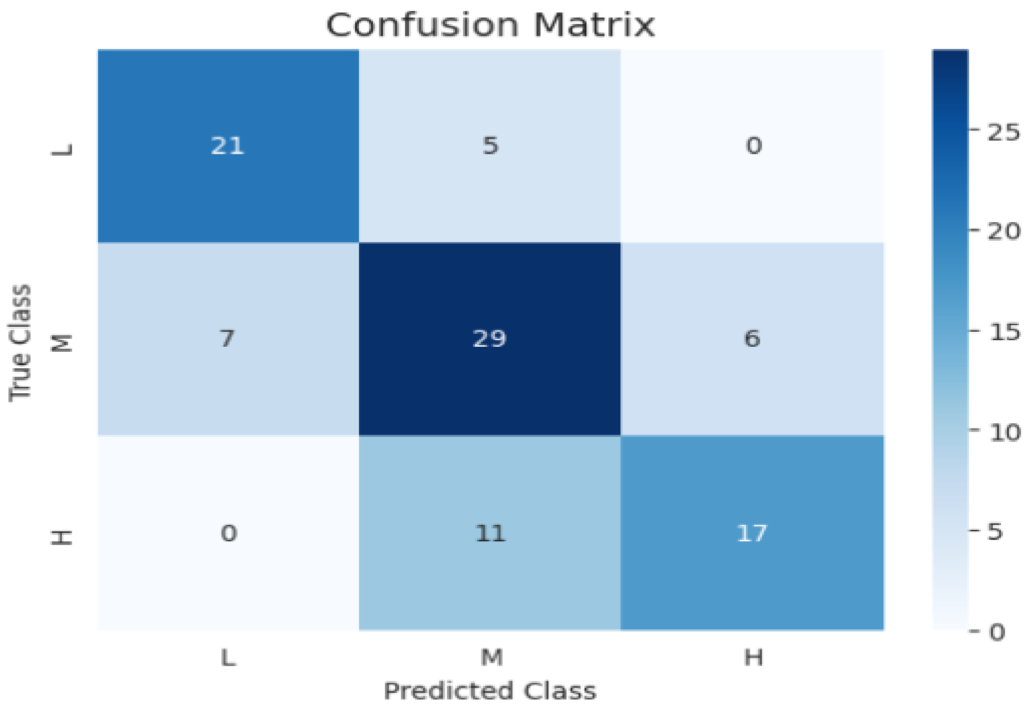
8. Model Accuracy Over Epochs

- **Chart Type:** Line Plot
- **Purpose:** Compare training and validation accuracy over epochs.
- **Insight:** Validates model convergence and generalization ability.



9.Confusion Matrix

- **Chart Type:** Heatmap
- **Purpose:** Show how well the model classifies each class (L, M, H).
- **Insight:** Medium class classified best; few misclassifications between adjacent classes.



10. Classification Report Table

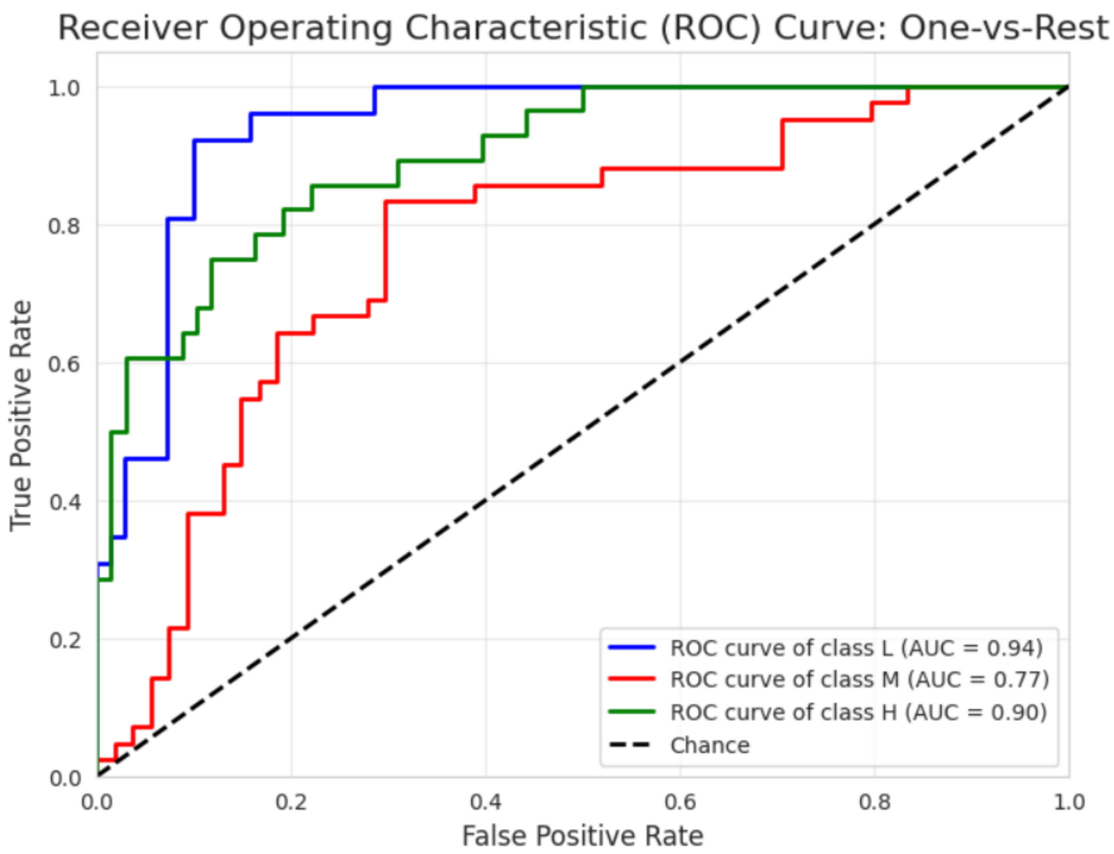
- **Chart Type:** Text Table
- **Purpose:** Display precision, recall, F1-score, and support for each class.
- **Insight:** Balanced model with overall accuracy ≈ 70%.

--- Classification Report (Precision, Recall, F1-Score) ---

	precision	recall	f1-score	support
Low (L)	0.75	0.81	0.78	26
Medium (M)	0.64	0.69	0.67	42
High (H)	0.74	0.61	0.67	28
accuracy			0.70	96
macro avg	0.71	0.70	0.70	96
weighted avg	0.70	0.70	0.70	96

11. ROC Curve (One-vs-Rest)

- **Chart Type:** Line Plot (multi-class ROC)
- **Purpose:** Evaluate class separability using ROC-AUC metric.
- **Insight:** High AUC values (≥0.90) indicate strong model discrimination.



DEEP LEARNING MODEL

Model Overview

- **Type:** Sequential Multi-Layer Perceptron (MLP)
- **Task:** Multi-class classification of student performance into:
    - 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	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	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 applied Exploratory Data Analysis (EDA) and Deep Learning Regression to predict student performance. Key findings include:

- **Significant predictors:** G1, G2, study time, and absences, showing strong correlation with final grades.
- **Model performance:** The Multi-Layer Perceptron (MLP) achieved high accuracy, demonstrating its effectiveness for regression in educational analytics.
- **Practical impact:** Insights from this study can help educational institutions identify at-risk students early and implement timely interventions.

Future Scope:

1. **Behavioral and Psychological Features:** Integrate variables such as motivation, stress, and engagement for richer predictive modeling.
2. **Ensemble Models:** Compare MLP performance with advanced models like **XGBoost** or **Gradient Boosting** for potentially improved accuracy.
3. **Interactive Dashboards:** Develop real-time dashboards for monitoring student performance and intervention planning.
4. **Larger Datasets:** Extend the study to **multi-school datasets** for better generalization.
5. **Explainable AI:** Incorporate **SHAP** or **LIME** to interpret model predictions and improve transparency for stakeholders.

REFERENCE

UCI Machine Learning Repository – Student Performance Dataset. https://archive.ics.uci.edu/dataset/320/student+performance

Géron, A. (2019). *Hands-On Machine Learning with Scikit-Learn, Keras, and TensorFlow*. O'Reilly Media.

Chollet, F. (2018). *Deep Learning with Python*. Manning Publications.

Raschka, S., & Mirjalili, V. (2019). *Python Machine Learning*. Packt Publishing.

McKinney, W. (2017). *Python for Data Analysis*. O’Reilly Media.

Pedregosa, F. et al. (2011). *Scikit-learn: Machine Learning in Python*. JMLR, 12, 2825–2830.

Hunter, J. D. (2007). *Matplotlib: A 2D Graphics Environment*. Computing in Science & Engineering, 9(3), 90–95.

APPENDIX(CODE SECTION)

```
# ============================
# Imports
# ============================
import os
import zipfile
import math
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt

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

try:
    import tensorflow as tf
    from tensorflow import keras
    from tensorflow.keras import layers
except Exception as e:
    raise ImportError(
        "TensorFlow is required to run the training. "
        "If you are in Colab, TF is preinstalled. "
        "Locally run: pip install tensorflow"
    ) from e

# ============================
# Load Dataset
# ============================
zip_path = "/mnt/data/student+performance.zip"
```

```
if not os.path.exists(zip_path):
    zip_path = "student+performance.zip"  # fallback if running locally
assert os.path.exists(zip_path), f"Zip file not found at {zip_path}."

extract_dir = "/mnt/data/student_performance_unzipped"
os.makedirs(extract_dir, exist_ok=True)

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

csv_files = [f for f in os.listdir(extract_dir) if f.lower().endswith(".csv")]
print("CSV files found:", csv_files)

# Pick student-mat.csv if exists else first CSV
csv_choice = None
for f in csv_files:
    if "mat" in f.lower():
        csv_choice = f
        break
if csv_choice is None:
    csv_choice = csv_files[0]

csv_path = os.path.join(extract_dir, csv_choice)
print("Using CSV:", csv_path)

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

# ==========================
# EDA & Cleaning
# ==========================
print(df.info())
display(df.describe().T)
```

```
# Missing values
miss = df.isnull().sum()
print("Missing per column (non-zero shown):")
print(miss[miss > 0])

# Duplicates
print("Duplicate rows count:", df.duplicated().sum())
if df.duplicated().sum() > 0:
    df = df.drop_duplicates()

# Outlier check using IQR for numeric columns
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 = Q1 - 1.5 * IQR
    upper = Q3 + 1.5 * IQR
    outlier_count = ((df[col] < lower) | (df[col] > upper)).sum()
    outlier_info[col] = outlier_count
print("Outlier counts (IQR):")
print(outlier_info)

# ===========================
# Visualizations
# ===========================
plt.figure(figsize=(6,4))
plt.hist(df['G3'], bins=11)
plt.title("Histogram of final grade (G3)")
plt.xlabel("G3")
plt.ylabel("Count")
plt.show()

# Correlation heatmap
```

```
corr = df[num_cols].corr()
plt.figure(figsize=(10,8))
plt.imshow(corr, interpolation='nearest', aspect='auto')
plt.colorbar()
plt.xticks(range(len(num_cols)), num_cols, rotation=90)
plt.yticks(range(len(num_cols)), num_cols)
plt.title("Correlation matrix (numeric features)")
plt.tight_layout()
plt.show()

# Boxplot of G3 by sex
plt.figure(figsize=(6,4))
sex_unique = list(df['sex'].unique())
data_by_sex = [df[df['sex']==s]['G3'] for s in sex_unique]
plt.boxplot(data_by_sex, labels=sex_unique)
plt.title("Boxplot of G3 by sex")
plt.xlabel("Sex")
plt.ylabel("G3")
plt.show()

# Scatter: G1 vs G3
plt.figure(figsize=(6,4))
plt.scatter(df['G1'], df['G3'])
plt.title("G1 vs G3 (first period grade vs final)")
plt.xlabel("G1")
plt.ylabel("G3")
plt.show()

# Bar chart: school counts
plt.figure(figsize=(6,4))
school_counts = df['school'].value_counts()
plt.bar(school_counts.index, school_counts.values)
plt.title("Count of students by school")
plt.xlabel("School")
plt.ylabel("Count")
```

```
plt.show()

# ===========================
# Preprocessing Pipeline
# ===========================
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=False))
])

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

X_processed = preprocessor.fit_transform(X)
print("Processed 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 = 0.15 of total -> val size w.r.t train_full = 0.15/0.85 ≈ 0.17647
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("Shapes -> train, val, test:", X_train.shape, X_val.shape,
X_test.shape)

# ===========================
# Build MLP Model
# ===========================
def build_mlp(input_dim, units=64, dropout_rate=0.0, 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_hist = None
best_params = None

for params in search_space:
    print("Training with:", params)
    model = build_mlp(input_dim, units=params['units'], lr=params['lr'],
dropout_rate=0.1)
    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("Val RMSE:", val_rmse)

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

print("Best params:", best_params, "Val RMSE:", best_val_rmse)

# ===========================
# Evaluate on Test Set
# ===========================
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"Test RMSE: {test_rmse:.4f}, Test MAE: {test_mae:.4f}, Test R2:
{test_r2:.4f}")

# ==========================
# Save Model and Pipeline
# ==========================
model_path = "student_mlp_regressor.h5"
best_model.save(model_path)

pipeline_path = "preprocessing_pipeline.joblib"
joblib.dump(preprocessor, pipeline_path)

print("Saved model to:", model_path)
print("Saved preprocessing pipeline to:", pipeline_path)

# ==========================
# Plots: Training Curves & Predictions
# ==========================
hist = best_hist.history

# Loss vs Epoch
plt.figure(figsize=(6,4))
plt.plot(hist['loss'], label='train_loss')
plt.plot(hist['val_loss'], label='val_loss')
plt.title("Loss (MSE) vs Epoch")
plt.xlabel("Epoch")
plt.ylabel("Loss (MSE)")
plt.legend()
plt.show()

# MAE vs Epoch
plt.figure(figsize=(6,4))
plt.plot(hist['mae'], label='train_mae')
```

```
plt.plot(hist['val_mae'], label='val_mae')
plt.title("MAE vs Epoch")
plt.xlabel("Epoch")
plt.ylabel("MAE")
plt.legend()
plt.show()

# Predicted vs Actual
plt.figure(figsize=(6,6))
plt.scatter(y_test, test_preds)
plt.plot([0,20],[0,20], color='red')
plt.title("Predicted vs Actual (Test)")
plt.xlabel("Actual G3")
plt.ylabel("Predicted G3")
plt.show()

# Residuals
residuals = y_test - test_preds
plt.figure(figsize=(6,4))
plt.hist(residuals, bins=20)
plt.title("Residuals distribution (Test)")
plt.xlabel("Residual")
plt.ylabel("Count")
plt.show()

plt.figure(figsize=(6,4))
plt.scatter(y_test, residuals)
plt.axhline(0, color='red')
plt.title("Residuals vs Actual G3")
plt.xlabel("Actual G3")
plt.ylabel("Residual")
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)
```