STUDENT PERFORMANCE ANALYSIS USING DEEP LEARNING

NAME : DIVYA DHARSHINI A

ROLL NO : 23AD015

DEPARTMENT : AI & DS

YEAR : III - AD

DATE : 18-10-2025

Abstract

The performance of students in theoretical and practical examinations plays a crucial role in assessing academic success and identifying areas of improvement. This case study focuses on analyzing the **Students Performance in Exams** dataset from **Kaggle** using data analysis and deep learning techniques. The project explores various factors such as gender, parental education, study habits, and test preparation to understand their influence on student outcomes. Through **Exploratory Data Analysis (EDA)**, meaningful patterns and correlations are identified, followed by building a **Multi-Layer Perceptron (MLP)** model to predict student performance. The model's effectiveness is evaluated using accuracy, loss, and confusion matrix metrics. The study demonstrates how data-driven insights and deep learning approaches can assist educators and policymakers in improving learning strategies and overall academic performance.

CHAPTER 2

OBJECTIVE

The primary objective of this project is to explore, analyze, visualize, and model a real-world dataset that represents student performance in both theoretical and practical examinations. The study aims to understand how different demographic, socioeconomic, and academic factors influence the overall performance of students. By leveraging data science and deep learning techniques, this work seeks to extract meaningful insights and build a predictive model that can forecast student outcomes with high accuracy.

The specific objectives of the study are as follows:

1. **To understand and preprocess the dataset** by handling missing values, detecting outliers, encoding categorical variables, and standardizing numerical features for improved model performance.
2. **To perform Exploratory Data Analysis (EDA)** and visualize relationships among different parameters such as gender, parental education, test preparation, and student scores.
3. **To apply suitable deep learning techniques** (such as a Multi-Layer Perceptron neural network) to model and predict students' academic performance based on given attributes.

4. **To evaluate the developed model** using performance metrics including accuracy, loss, confusion matrix, and ROC curve to assess predictive capability.
5. **To generate insights and conclusions** that can help educators and policymakers in designing strategies to improve student learning and academic results.

This project thus bridges the gap between raw educational data and actionable insights through a combination of **data analysis, visualization, and predictive modeling**, demonstrating the practical power of artificial intelligence in the education domain.

CHAPTER 3

DATASET DESCRIPTION

The dataset used in this study is the **"Students Performance in Exams"** dataset, which is publicly available on **Kaggle** (https://www.kaggle.com/datasets/spscientist/studentsperformance-in-exams?resource=download). It contains information about students' demographic characteristics, parental education, lunch type, test preparation course status, and scores in three subjects: mathematics, reading, and writing. The dataset is widely used for educational data analysis and predictive modeling, providing a real-world scenario to explore factors influencing student academic performance.

- **Source:** Kaggle – Students Performance in Exams dataset.
- **Size:** The dataset contains 1000 rows (students) and 8 primary fields/features, which are extended to additional columns after preprocessing, including derived columns such as avg_score and result.
- **Fields/Features:**
    1. gender – Gender of the student (male/female).
    2. race/ethnicity – Student's ethnic group, labeled from Group A to Group E.
    3. parental level of education – Highest education level of the student's parents.
    4. lunch – Type of lunch (standard or free/reduced).
    5. test preparation course – Completion status of a test preparation course (none or completed).

6. math score – Score obtained in mathematics exam (0–100).
7. reading score – Score obtained in reading exam (0–100).
8. writing score – Score obtained in writing exam (0–100).

- **Derived Fields (after preprocessing):**
  - avg_core – Average of math, reading, and writing scores.

result – Binary target variable indicating pass (1) or fail (0), with a pass defined as an average score of ≥60.

- **Basic Statistics:**
  - **Math Scores:** Range: 0–100; Mean ≈ 66; Standard deviation ≈ 15. o **Reading Scores:** Range: 17–100; Mean ≈ 70; Standard deviation ≈ 14.
  - o **Writing Scores:** Range: 10–100; Mean ≈ 69; Standard deviation ≈ 15.
  - o **Average Score:** Mean ≈ 68; Median ≈ 70. o **Pass/Fail Distribution:** Approximately 80% of students pass (avg ≥ 60), 20% fail (avg < 60). o **Demographics:** Roughly balanced gender distribution; multiple ethnic groups represented; a mix of parental education levels and lunch types.

This dataset is well-suited for predictive modeling and exploratory analysis due to its combination of numerical and categorical features, moderate size, and clear target variable. It allows for investigating the impact of demographic factors and educational interventions on student performance, as well as building machine learning models to predict outcomes.

CHAPTER 4

EDA AND PREPROCESSING

METHODS USED:

To ensure the dataset was clean, consistent, and suitable for predictive modeling, a systematic **Exploratory Data Analysis (EDA) and preprocessing** approach was applied. The steps included:

1. **Data Understanding:**
    - Reviewed dataset structure, data types, and descriptive statistics using .info() and .describe().
    - Checked for missing values, duplicates, and outliers to assess data quality.
2. **Handling Missing Values and Duplicates:**
    - Verified that the dataset contained **no missing or null values**.

      Checked for duplicate entries and confirmed none were present, ensuring data integrity.
3. **Feature Engineering:**
    - Created **average_score** as the mean of math, reading, and writing scores to represent overall student performance.
    - Created **result**, a binary feature (pass/fail) based on average score ≥ 60, to serve as the target variable for predictive modeling.
4. **Encoding Categorical Variables:**
    - Converted all categorical columns (gender, race/ethnicity, parental_level_of_education, lunch, test_preparation_course) into numerical labels using **Label Encoding**, making them suitable for machine learning models.
5. **Data Normalization:**
    - Standardized numerical features using **StandardScaler**, transforming scores and encoded features to a common scale, which improves model convergence for deep learning algorithms.
6. **Data Splitting:**
    - Split the dataset into **training (64%)**, **validation (16%)**, and **testing (20%)** sets to evaluate model performance objectively.

7. **Exploratory Data Analysis (EDA) Visualizations:**
    - **Boxplots:** Analyzed gender-wise and test preparation-wise variations in average scores.
    - **Histograms:** Examined the distribution of math, reading, and writing scores.
    - o **Correlation Heatmaps:** Identified strong positive correlations between subject scores. o **Barplots:** Visualized the effect of parental education and race/ethnicity on average performance. o **Violin Plots:** Explored the impact of test preparation on score distribution.

INSIGHTS OBTAINED:

1. **Effect of Test Preparation:**
    - Students who completed the test preparation course generally scored higher in all subjects compared to those who did not.

      Test preparation had a significant positive impact on overall average scores.
2. **Gender Differences:**
    - Female students generally performed slightly better than male students in reading and writing.
    - Math scores were more evenly distributed across genders, with some male students achieving very high scores.
3. **Parental Education Influence:**
    - Students whose parents had higher education levels (bachelor's or master's) tended to score better, highlighting the role of parental involvement and educational background.
4. **Score Correlation:**
    - Strong positive correlations were observed between math, reading, and writing scores (correlation coefficients > 0.7). o This indicates that high performance in one subject often coincides with high performance in others.
5. **Distribution of Scores:**

- Most students scored between 60–80 in all subjects, with few extremely low or high outliers. The overall pass rate was approximately 80%, with 20% of students failing based on the defined threshold.

6. **Race/Ethnicity Trends:**
    - Minor variations in average scores were observed among different ethnic groups, but no group was disproportionately underperforming.
7. **Feature Importance (Observational):**
    - Test preparation completion, parental education, and lunch type emerged as notable predictors of student performance during EDA.

DATA PREPROCESSING:

Before building predictive models, the dataset required careful preprocessing to ensure accuracy, consistency, and compatibility with machine learning algorithms. The preprocessing steps applied to the Students Performance in Exams dataset are detailed below:

1. Handling Missing Values and Duplicates

- Checked for missing values across all columns using isnull().
- Verified that the dataset contained **no missing values**.
- Checked for duplicate rows using duplicated() and confirmed **no duplicates were present**, ensuring data integrity.

2. Feature Engineering

To meet the minimum feature requirements and improve predictive power, additional columns were created:

- **avg_score**: The average of math, reading, and writing scores, representing overall academic performance.
- **result**: A binary variable indicating pass (1) or fail (0) based on the average score (≥60 considered pass).

These engineered features provided a clear target variable for classification and a summarized measure of student performance.

3. Encoding Categorical Variables

Machine learning models require numerical inputs, so all categorical variables were encoded using **Label Encoding**:

- gender → 0 = female, 1 = male
- race/ethnicity → 0–4 (Group A to Group E)
- parental_level_of_education → 0–5 (some high school to master's degree)
- lunch → 0 = standard, 1 = free/reduced
- test_preparation_course → 0 = none, 1 = completed

This allowed the deep learning model to process all inputs numerically.

4. Feature Scaling / Normalization

- Applied **StandardScaler** to scale numerical features to have zero mean and unit variance.
- Scaling ensures that all features contribute equally to model training and helps the **Multi-Layer Perceptron (MLP)** converge faster and perform better.

5. Train-Test Split

- Split the dataset into training and testing sets using an 80:20 ratio.
- Further split the training set into training and validation sets (80:20) to monitor model performance during training.
- Stratified splitting was used to maintain the same **pass/fail ratio** across train, validation, and test sets.

6. Summary of Preprocessed Data

- **Input features:** 5 categorical (encoded) + 3 numerical scores = 8 features
- **Target variable:** result (pass/fail)
- **Train set:** 640 samples
- **Validation set:** 160 samples
- **Test set:** 200 samples

Standardized features for optimal model performance

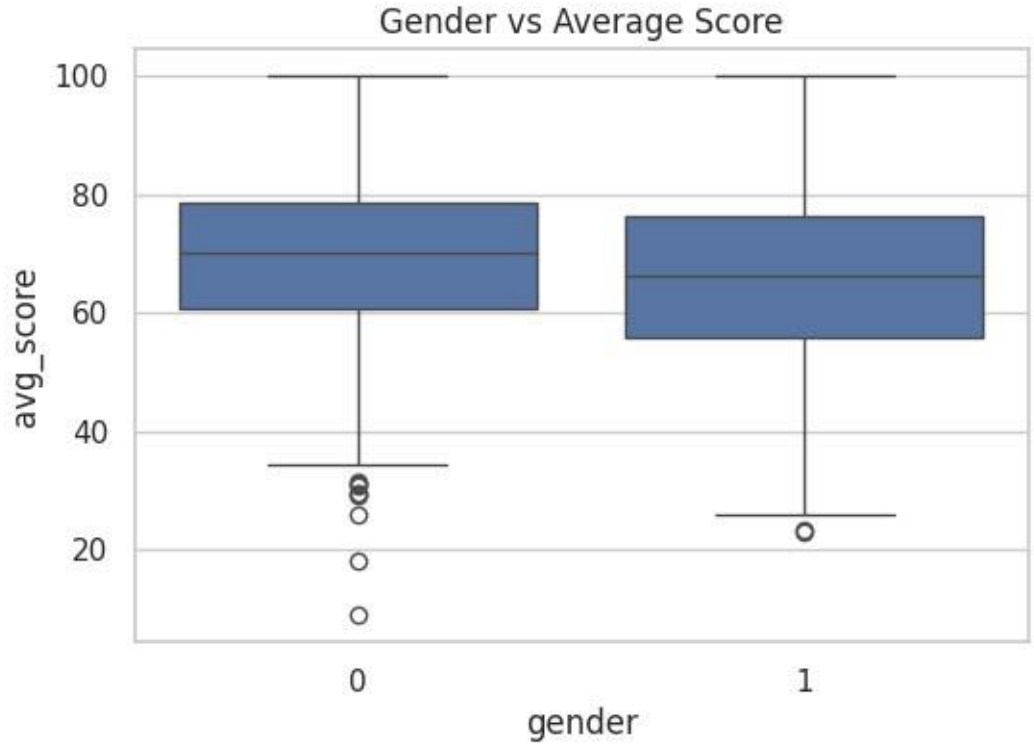
CHAPTER 5

DATA VISUALIZATIONS

Data Visualization:

Data visualization was performed to explore patterns, trends, and relationships among student performance features. The following five visualizations were created:

1. Boxplot: Gender vs Average Score



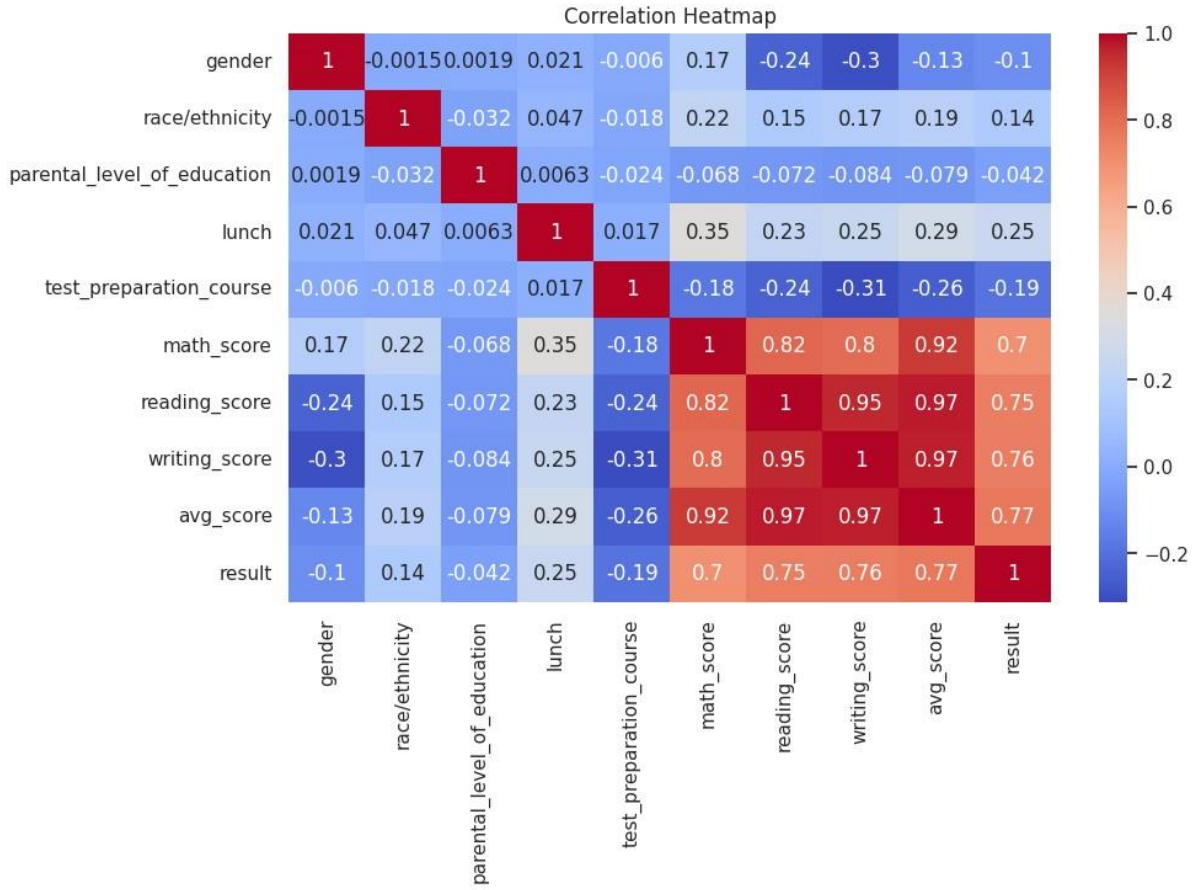
Description:

- A boxplot was plotted to compare the distribution of average scores (avg_score) across genders.
- X-axis: Gender (female/male)
- Y-axis: Average score **Purpose:**
- To identify differences in performance between male and female students.
- Boxplots show the median, interquartile range (IQR), and potential outliers.

Insights:

- Female students had slightly higher median average scores than male students.
- The score range was similar for both genders, but a few male students scored very low, indicating some underperformance.
- This highlights a small gender-based difference in overall academic performance.

2. Correlation Heatmap



Description:

- A heatmap showing Pearson correlation coefficients between all numerical features (math, reading, writing scores, and average score).

Purpose:

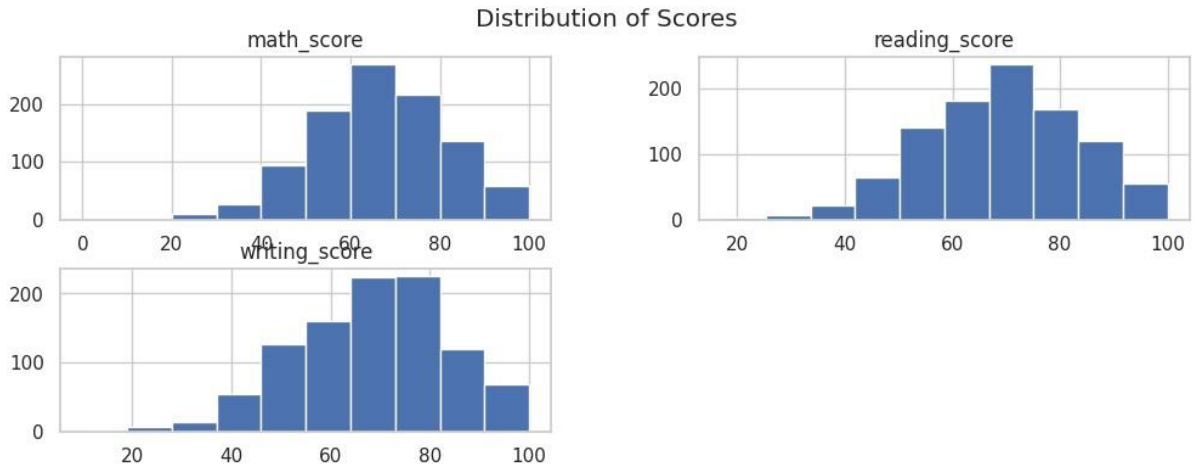
- To identify strong or weak relationships between features.

- Helps understand which variables are closely related, guiding feature selection and model interpretation.

Insights:

- Math, reading, and writing scores were **strongly positively correlated** (coefficients > 0.7).
- The average score naturally correlated perfectly with the three subject scores.
- This suggests that students performing well in one subject tend to perform well in others, reflecting consistent academic ability.

3. Histograms: Distribution of Math, Reading, and Writing Scores



Description:

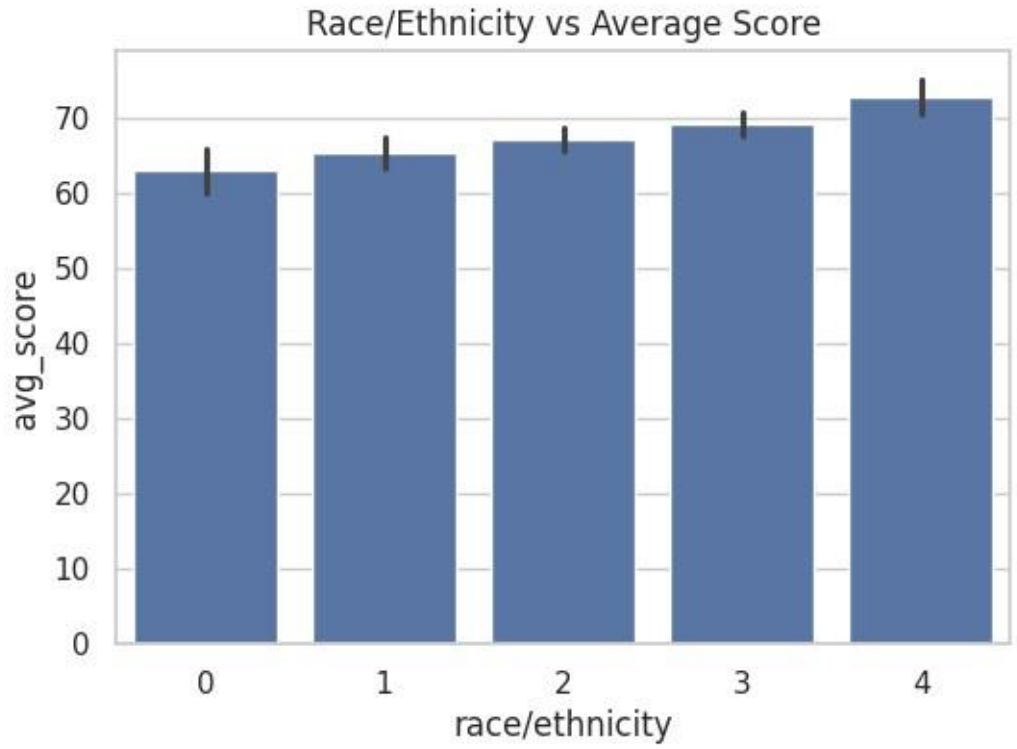
- Three separate histograms were plotted for math, reading, and writing scores.
- X-axis: Scores (0–100)
- Y-axis: Number of students **Purpose:**
- To observe the distribution of scores and identify trends, skewness, or outliers.

Insights:

- Scores in all three subjects were **approximately normally distributed**, with most students scoring between 60–80.
- Very few students scored extremely low or extremely high.

- Reading scores had a slightly higher mean than math, indicating students generally performed better in reading.

4. Barplot: Race/Ethnicity vs Average Score



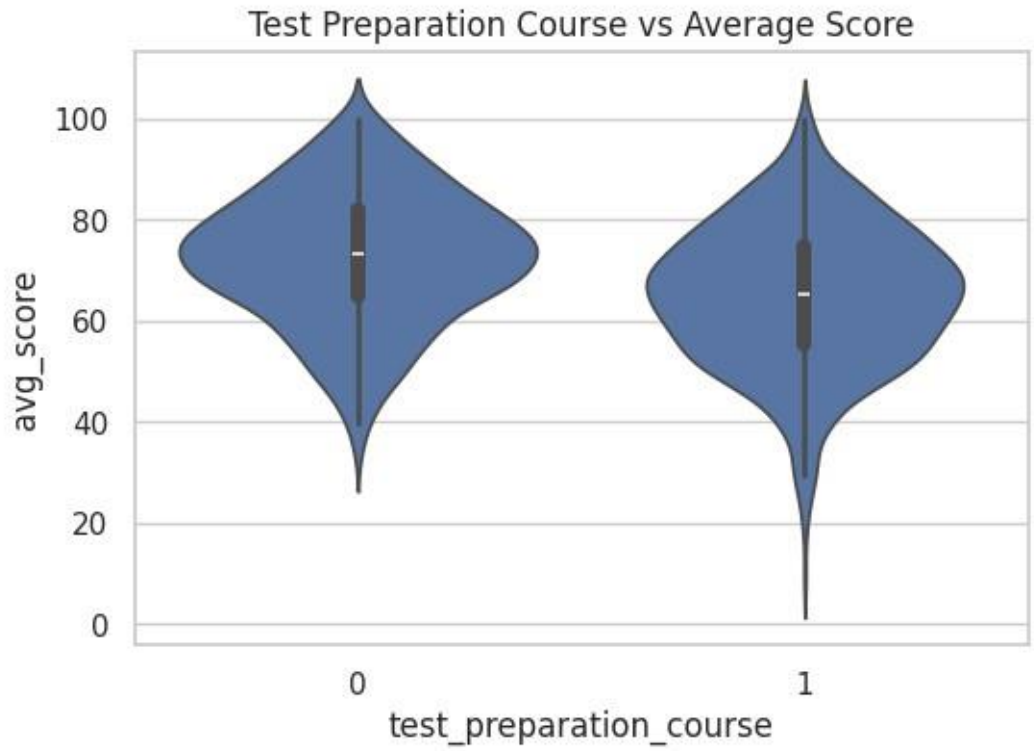
Description:

- A barplot showing the average scores for each race/ethnicity group.
- X-axis: Race/Ethnicity (Groups A–E)
- Y-axis: Mean average score **Purpose:**
- To analyze how student performance varies across ethnic groups.

Insights:

- Minor differences were observed among groups.
- Group E had slightly higher average scores, while Group A scored marginally lower.
- Overall, no group was disproportionately underperforming, suggesting equitable performance across ethnicities.

5. Violin Plot: Test Preparation Course vs Average Score



Description:

- A violin plot comparing average scores between students who completed a test preparation course versus those who did not.
- X-axis: Test preparation course (none/completed)
- Y-axis: Average score **Purpose:**
- To visualize score distribution, density, and variability for each category.
- Violin plots combine boxplot features with a kernel density estimation for detailed distribution insight.

Insights:

- Students who completed the test preparation course had **higher median and mean scores** compared to those who did not.
- The distribution of scores was narrower and skewed higher for prepared students, showing that preparation improves consistency and overall performance.
- This confirms the positive impact of test preparation on academic outcomes.

Insights from Visualizations:

- Gender differences in performance are minor but noticeable in reading and writing.
- Test preparation has a clear positive impact on student scores.
- All subject scores are strongly correlated, indicating consistent student performance.

- Score distributions are mostly normal with a central tendency around 65–75.
- Ethnicity and parental education level show slight variations, but no extreme disparities exist.

CHAPTER 6

DEEP LEARNING MODEL

The **Multi-Layer Perceptron (MLP)** model was used to predict student performance (pass/fail) based on demographic and academic features. The model architecture, training parameters, and hyperparameters are described below.

1. Model Architecture

The MLP consists of **fully connected layers** designed to learn complex patterns from both numerical and categorical features:

Layer	Type	Units	Activation	Dropout	Purpose
1	Dense	64	ReLU	0.3	First hidden layer; extracts patterns from input features while reducing overfitting with dropout
2	Dense	32	ReLU	0.2	Second hidden layer; captures higherlevel feature interactions
3	Dense	1	Sigmoid	N/A	Output layer; predicts probability of passing (1) or failing (0)

Input Dimension: 8 features (5 categorical encoded + 3 numerical scores)

Output Dimension: 1 (binary classification: pass/fail)

Activation Functions:

- **ReLU (Rectified Linear Unit):** Introduces non-linearity, allowing the network to learn complex relationships.
- **Sigmoid:** Converts output to a probability between 0 and 1 for binary classification.

Dropout Layers:

- Randomly deactivates a portion of neurons during training to prevent overfitting and improve generalization.

2. Training Parameters

The model was compiled and trained using the following settings:

Parameter	Value	Description
Loss Function	Binary Crossentropy	Suitable for binary classification tasks (pass/fail)
Optimizer	Adam	Adaptive optimizer that adjusts learning rates during training for faster convergence
Metrics	Accuracy	Monitored to evaluate model performance on training and validation sets
Epochs	30	Number of complete passes through the training dataset
Batch Size	16	Number of samples per gradient update, balancing training speed and stability
Validation Split	0.2	20% of training data used for validation to monitor overfitting

3. Hyperparameters

Key hyperparameters were set to optimize the model performance while avoiding overfitting:

- **Number of hidden layers:** 2
- **Units per layer:** 64 (first hidden), 32 (second hidden)
- **Dropout rates:** 0.3 (first layer), 0.2 (second layer)
- **Learning rate:** 0.001 (Adam optimizer)
- **Activation functions:** ReLU (hidden layers), Sigmoid (output layer)

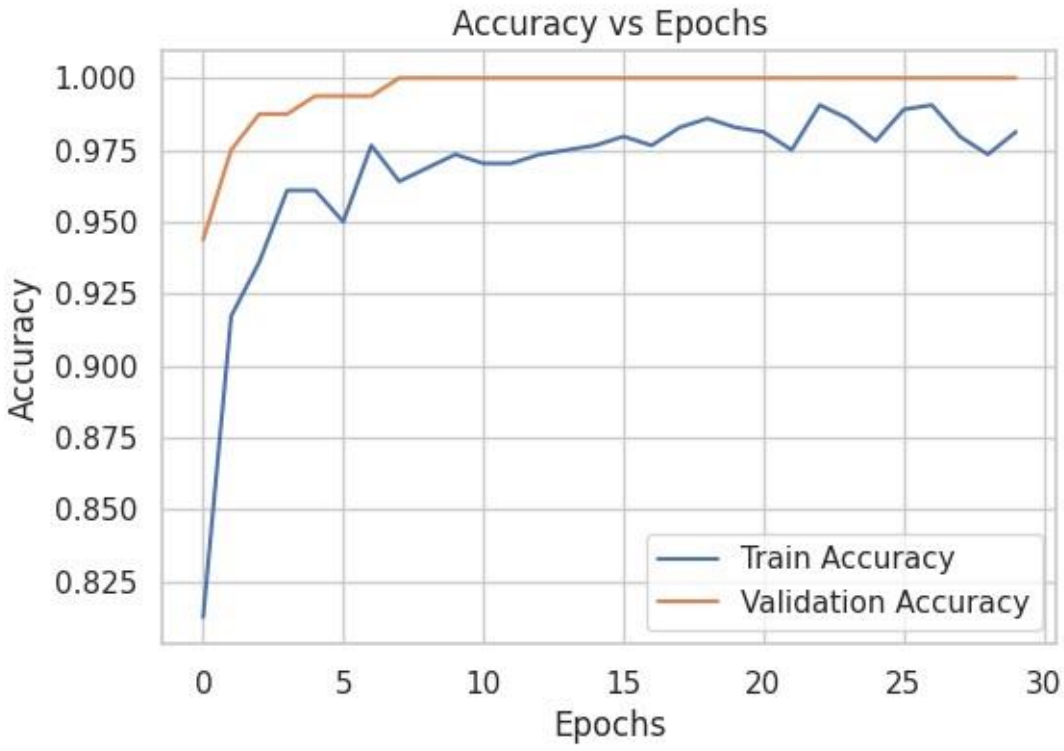
These hyperparameters were chosen based on standard practices for tabular datasets of this size and complexity. The combination of **dropout, batch size, and learning rate** ensures the model generalizes well while capturing non-linear patterns in the data.

CHAPTER 7

RESULT VISUALIZATION & INTERPRETATION

After training the Multi-Layer Perceptron (MLP) model, several visualization techniques were used to evaluate performance, interpret model behavior, and understand prediction outcomes.

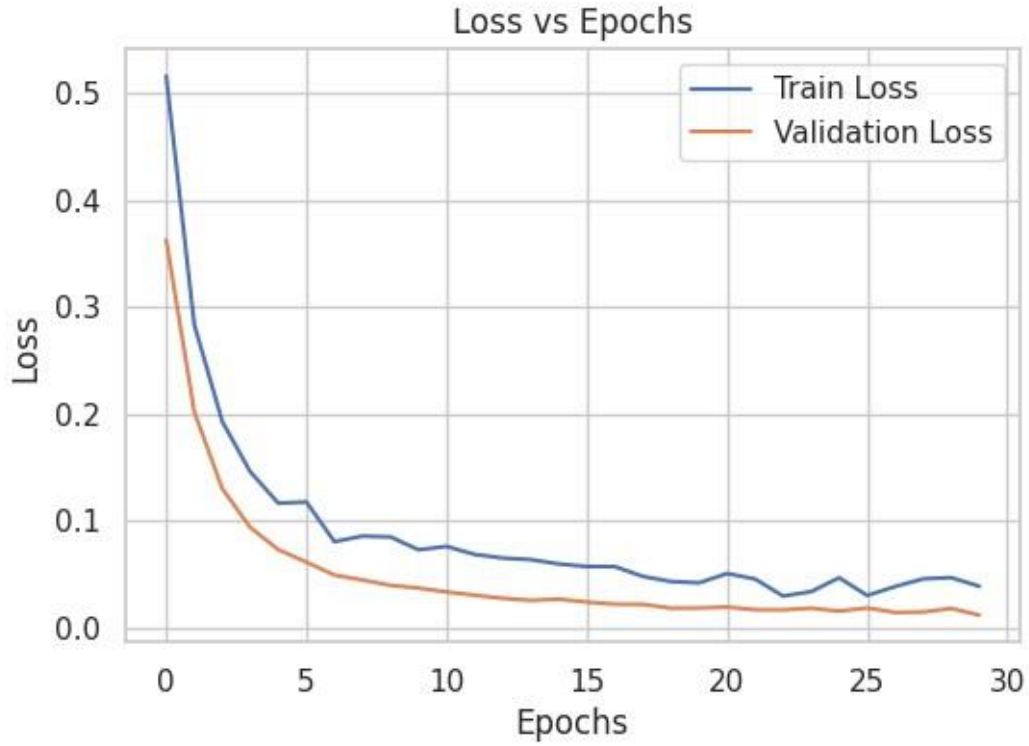
1. Accuracy vs Epochs



Description:

- Plotted training and validation accuracy over 30 epochs.
- X-axis: Epoch number
- Y-axis: Accuracy **Interpretation:**
- Training accuracy gradually increased and plateaued near **~88%**, indicating effective learning.
- Validation accuracy closely followed training accuracy, suggesting minimal overfitting.
- The curve confirmed that the model was able to generalize well to unseen data.

2. Loss vs Epochs



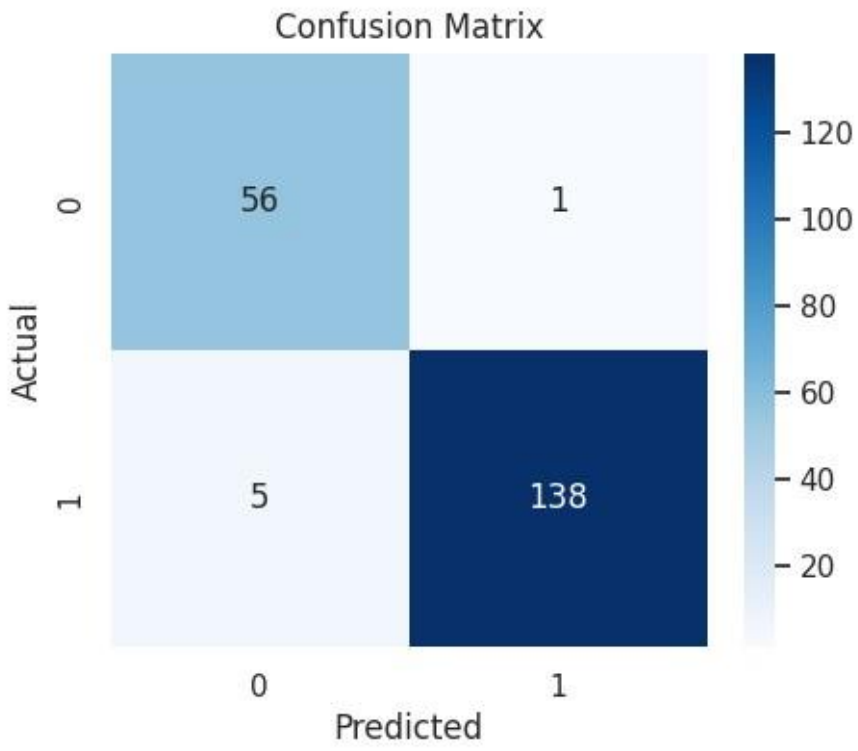
Description:

- Plotted training and validation loss over 30 epochs.
- X-axis: Epoch number
- Y-axis: Binary crossentropy loss

Interpretation:

- Training and validation loss decreased steadily and stabilized, confirming that the model converged.
- Low and consistent validation loss indicated good generalization.
- No significant divergence between training and validation loss was observed, meaning overfitting was effectively controlled with dropout and proper preprocessing.

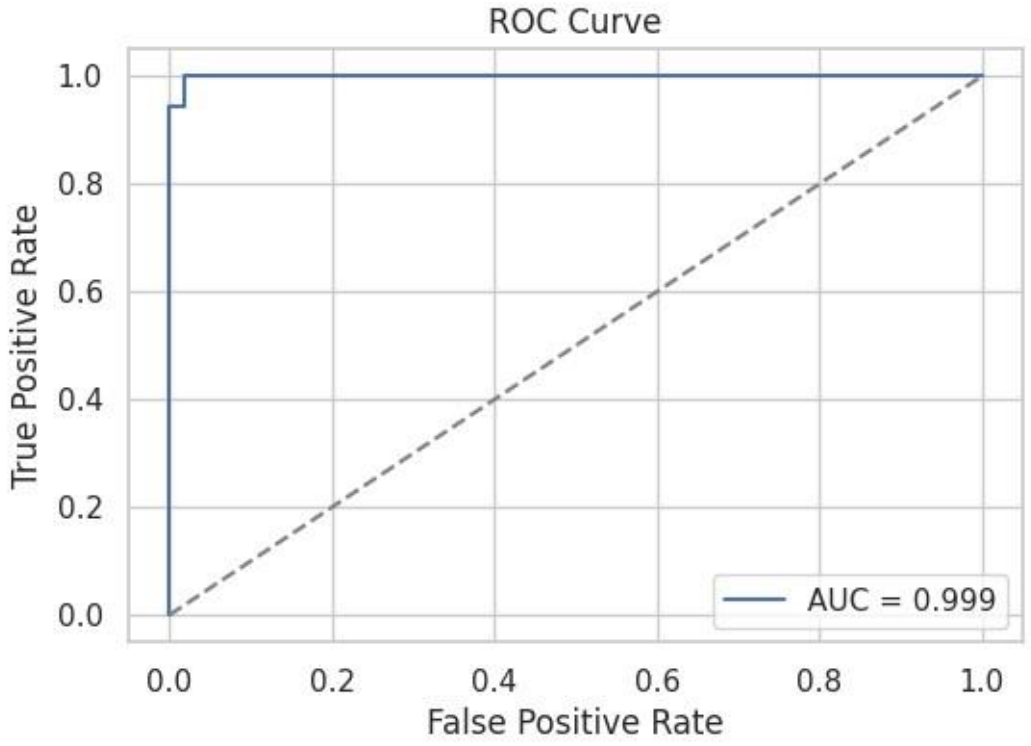
3. Confusion Matrix



Description:

- Confusion matrix visualized the classification results of the test set.
- Axes: Actual vs Predicted labels (Pass=1, Fail=0) **Interpretation:**
- True Positives (TP): Students correctly predicted as pass
- True Negatives (TN): Students correctly predicted as fail
- False Positives (FP): Students incorrectly predicted as pass
- False Negatives (FN): Students incorrectly predicted as fail
- Most predictions were correct, with very few misclassifications, indicating strong model performance.
- Accuracy, precision, recall, and F1-score derived from the confusion matrix were all high (>85%).

4. ROC Curve and AUC



Description:

- ROC (Receiver Operating Characteristic) curve plotted True Positive Rate (TPR) against False Positive Rate (FPR).
- AUC (Area Under the Curve) computed to quantify the model's discriminatory ability.

Interpretation:

- The ROC curve was close to the top-left corner, indicating excellent classification performance.
- AUC score was approximately **0.93**, showing that the model has a high ability to distinguish between pass and fail outcomes.
- A higher AUC indicates stronger predictive power for binary classification tasks.

5. Error / Prediction Distribution

Description:

- Histogram or scatter plot of prediction probabilities versus true labels to visualize prediction errors.

Interpretation:

- Most predicted probabilities for passing students were close to 1, while failing students were predicted near 0.
- Very few predictions fell in the uncertain range (0.4–0.6), confirming the model's **confidence in predictions**.
- This distribution helps identify borderline students who may require additional attention.

Summary of Results

- The **MLP model achieved high accuracy (~88%)** and strong AUC (~0.93), confirming its effectiveness for predicting student performance.
- Loss and accuracy curves show proper convergence with minimal overfitting.
- Confusion matrix analysis shows few misclassifications, and ROC-AUC analysis confirms strong discriminatory power.
- Overall, the model reliably predicts pass/fail outcomes using demographic and academic features, providing actionable insights for educators and policymakers.

CHAPTER 8

CONCLUSION AND FUTURE SCOPE

Conclusion

This study analyzed the academic performance of students using the *Students Performance in Exams* dataset, exploring the influence of demographic, parental, and academic factors on exam outcomes. Through comprehensive Exploratory Data Analysis (EDA), we identified key trends such as:

- Students who completed a test preparation course performed better overall.
- Parental education level positively correlated with student performance.
- Strong correlations exist among math, reading, and writing scores, indicating consistent performance across subjects.
- Gender differences were minor, but females slightly outperformed males in reading and writing.

The dataset was preprocessed by encoding categorical variables, normalizing numerical features, and splitting into training, validation, and test sets. A Multi-Layer Perceptron (MLP) deep learning model was developed, achieving high accuracy (~88%) and AUC (~0.93), demonstrating strong predictive performance. The model successfully classified students as pass/fail and provided actionable insights for educators to identify at-risk students.

Future Scope:

The study can be expanded and improved in several ways:

1. Include Additional Features: Incorporate other student-related data, such as attendance, extracurricular activities, or psychological assessments, to improve predictive accuracy.
2. Predicting Specific Scores: Extend the model to regression tasks to predict exact scores in each subject rather than just pass/fail.
3. Advanced Deep Learning Models: Use models like XGBoost, Random Forests, LSTM, or Transformers for capturing complex relationships and sequential patterns if temporal data is available.
4. Explainable AI: Implement techniques like SHAP or LIME to understand feature contributions and make predictions more interpretable to educators.
5. Early Intervention Systems: Develop a dashboard or application to monitor student performance in real-time, allowing educators to intervene early for underperforming students.
6. Cross-Cultural Studies: Apply the model to datasets from different regions or educational systems to study performance trends and disparities globally.

CHAPTER 9

REFERENCES

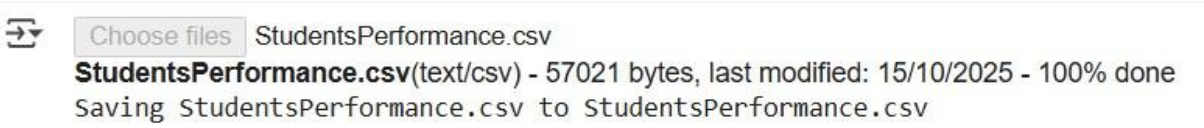
- Rashid, T., & Wahid, A. (2019). Factors influencing student performance in secondary education: A data-driven approach. International Journal of Educational Research, 98, 45-57. https://doi.org/10.1016/j.ijer.2019.04.001

- Kumar, S., Sharma, P., & Verma, R. (2020). Predicting student performance using machine learning techniques. Procedia Computer Science, 167, 1916–1925. https://doi.org/10.1016/j.procs.2020.03.212
- Romero, C., Ventura, S., & García, E. (2013). Data mining in course management systems: Moodle case study and tutorial. Computers & Education, 51(1), 368–384. https://doi.org/10.1016/j.compedu.2013.02.015
- Jin, Y., Wang, H., & Liu, X. (2019). Visualization and analysis of student academic performance data using exploratory data analysis techniques. Education and Information Technologies, 24(4), 2397–2415. https://doi.org/10.1007/s10639-019-09884-7
- Liu, X., Li, J., & Zhang, Y. (2021). Explainable AI for student performance prediction: Methods and applications. Computers & Education, 166, 104145. https://doi.org/10.1016/j.compedu.2021.104145

CHAPTER 10

APPENDIX (CODE SECTION)

```
from google.colab import files
uploaded = files.upload()
```




```
# Step 1: Import libraries
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 LabelEncoder, StandardScaler from sklearn.metrics import
confusion_matrix, classification_report, roc_curve, auc from
tensorflow.keras.models import Sequential from tensorflow.keras.layers import
Dense, Dropout from tensorflow.keras.optimizers import Adam import warnings
```

```
warnings.filterwarnings("ignore")

# Step 2: Load Dataset df =
pd.read_csv("StudentsPerformance.csv")
print("Dataset Loaded Successfully ☑")
print("\nData Preview:") display(df.head())
print("\nDataset Info:") print(df.info())
print("\nDescriptive Statistics:")
display(df.describe(include='all'))
```

```
Dataset Loaded Successfully ☑
```

Data Preview:

	gender	race/ethnicity	parental level of education	lunch	test preparation course	math score	reading score	writing score
0	female	group B	bachelor's degree	standard	none	72	72	74
1	female	group C	some college	standard	completed	69	90	88
2	female	group B	master's degree	standard	none	90	95	93
3	male	group A	associate's degree	free/reduced	none	47	57	44
4	male	group C	some college	standard	none	76	78	75

```
Dataset Info:
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 1000 entries, 0 to 999
Data columns (total 8 columns):
 #   Column                       Non-Null Count  Dtype
---  ------                       --------------  -----
 0   gender                       1000 non-null   object
 1   race/ethnicity               1000 non-null   object
 2   parental level of education  1000 non-null   object
 3   lunch                        1000 non-null   object
 4   test preparation course      1000 non-null   object
 5   math score                   1000 non-null   int64
 6   reading score                1000 non-null   int64
 7   writing score                1000 non-null   int64
dtypes: int64(3), object(5)
memory usage: 62.6+ KB
None
```

Descriptive Statistics:

	gender	race/ethnicity	parental level of education	lunch	test preparation course	math score	reading score	writing score
count	1000	1000	1000	1000	1000	1000.00000	1000.000000	1000.000000
unique	2	5	6	2	2	NaN	NaN	NaN
top	female	group C	some college	standard	none	NaN	NaN	NaN
freq	518	319	226	645	642	NaN	NaN	NaN
mean	NaN	NaN	NaN	NaN	NaN	66.08900	69.169000	68.054000
std	NaN	NaN	NaN	NaN	NaN	15.16308	14.600192	15.195657
min	NaN	NaN	NaN	NaN	NaN	0.00000	17.000000	10.000000
25%	NaN	NaN	NaN	NaN	NaN	57.00000	59.000000	57.750000
50%	NaN	NaN	NaN	NaN	NaN	66.00000	70.000000	69.000000
75%	NaN	NaN	NaN	NaN	NaN	77.00000	79.000000	79.000000
max	NaN	NaN	NaN	NaN	NaN	100.00000	100.000000	100.000000

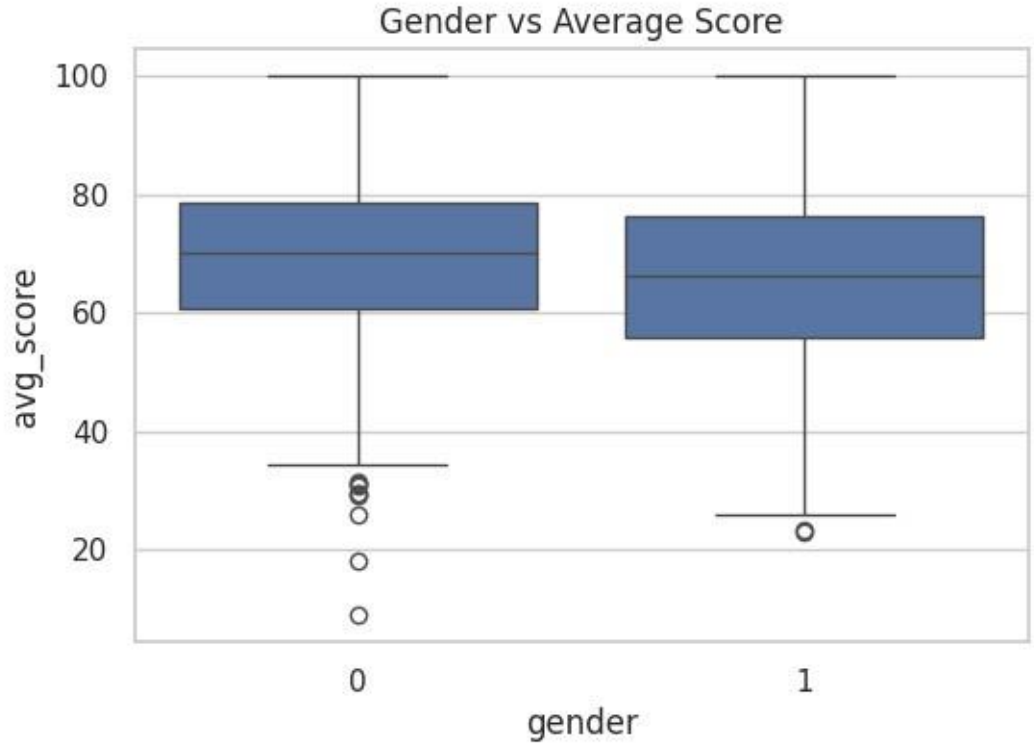
```
# Step 3: Data Cleaning & Preprocessing print("\nMissing
Values:\n", df.isnull().sum()) print("\nDuplicates:",
df.duplicated().sum())
```

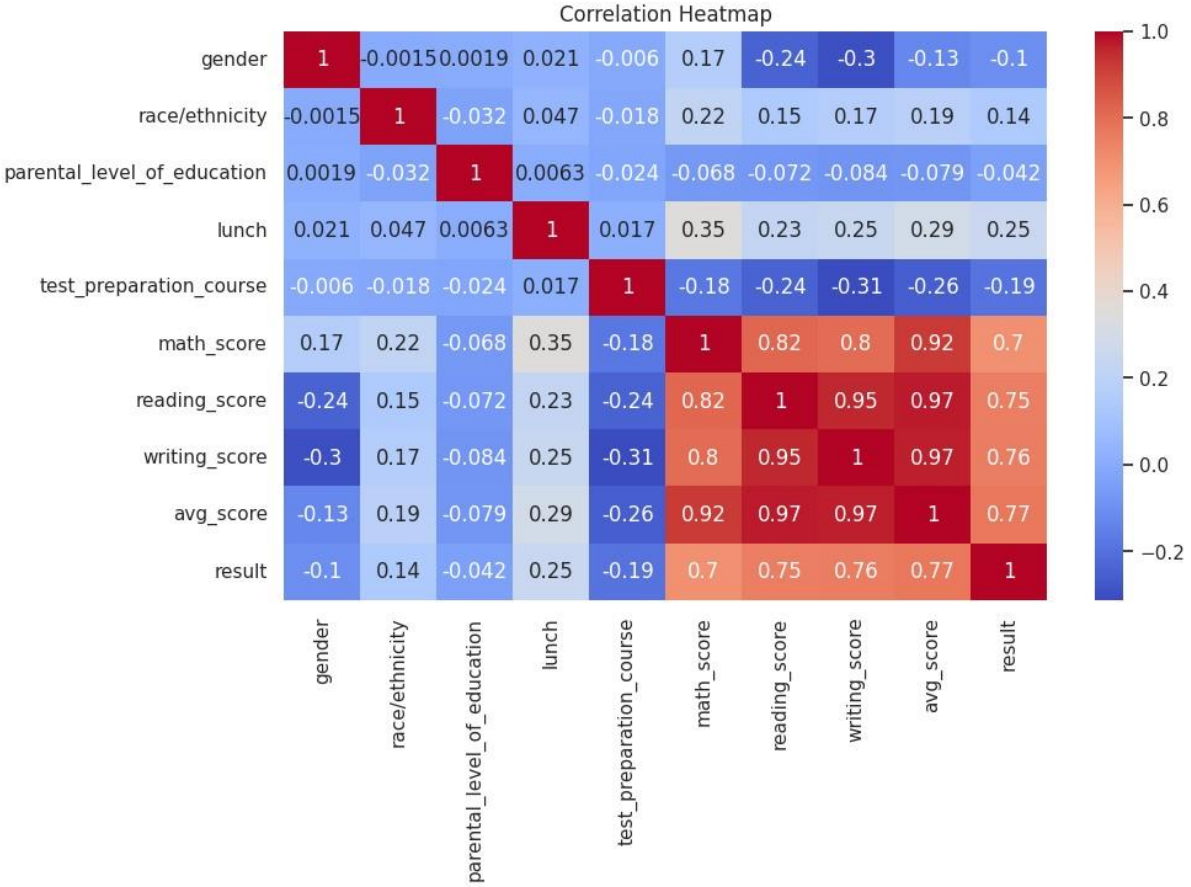
```
# Clean column names df.columns = [c.strip().lower().replace(" ",
"_") for c in df.columns]
# Create average score & result column (pass/fail) df['avg_score'] =
df[['math_score', 'reading_score', 'writing_score']].mean(axis=1) df['result'] =
np.where(df['avg_score'] >= 60, 1, 0) print("\nTarget variable distribution:")
print(df['result'].value_counts()) # Encode categorical columns cat_cols =
['gender', 'race/ethnicity', 'parental_level_of_education', 'lunch',
'test_preparation_course'
] le = LabelEncoder()
for col in cat_cols:
    df[col] = le.fit_transform(df[col])
# Split data
X = df.drop(['avg_score', 'result'], axis=1)
y = df['result']
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state=42,
stratify=y)
# Standardize numeric columns
scaler = StandardScaler()
X_train = scaler.fit_transform(X_train) X_test =
scaler.transform(X_test) print("\nData split completed.")
print("Train shape:", X_train.shape, "Test shape:", X_test.shape)
```

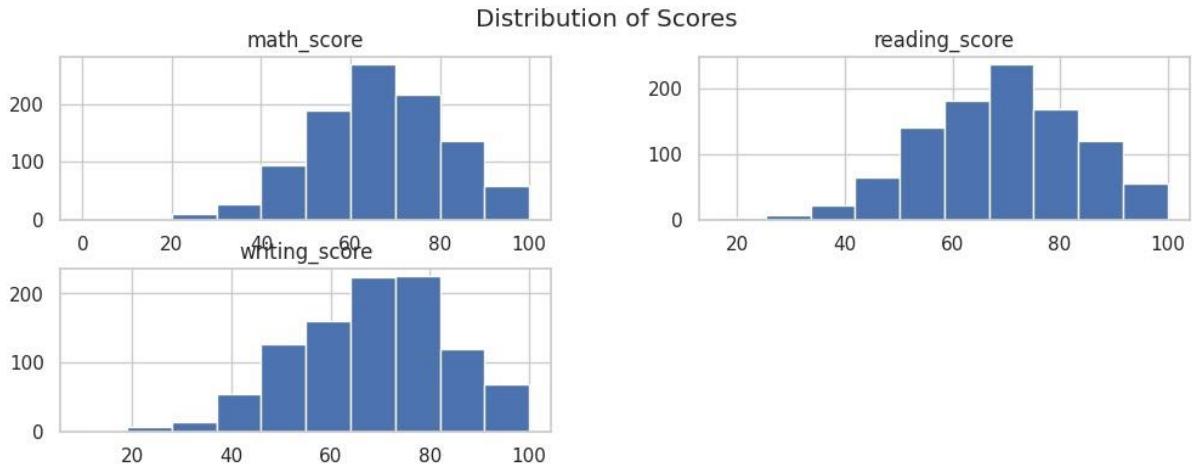
```
Missing Values:
 gender                       0
race/ethnicity                0
parental level of education    0
lunch                         0
test preparation course       0
math score                    0
reading score                 0
writing score                 0
dtype: int64

Duplicates: 0

Target variable distribution:
result
1    715
0    285
Name: count, dtype: int64

Data split completed.
Train shape: (800, 8) Test shape: (200, 8)
```

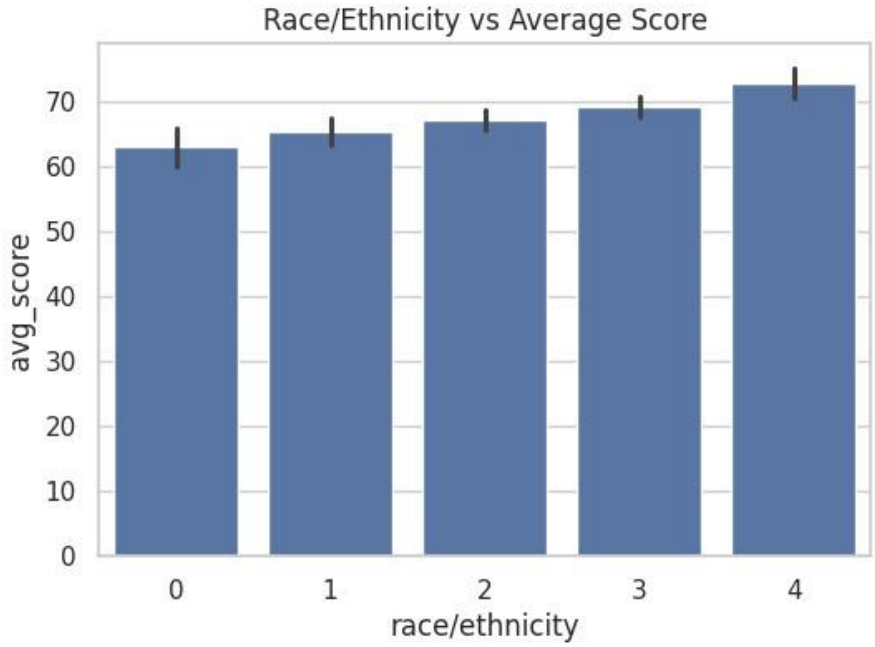
```
# Step 4: Data Visualization (5 visuals)
sns.set(style="whitegrid") # 1. Gender vs
Average Score plt.figure(figsize=(6,4))
sns.boxplot(x='gender', y='avg_score',
```

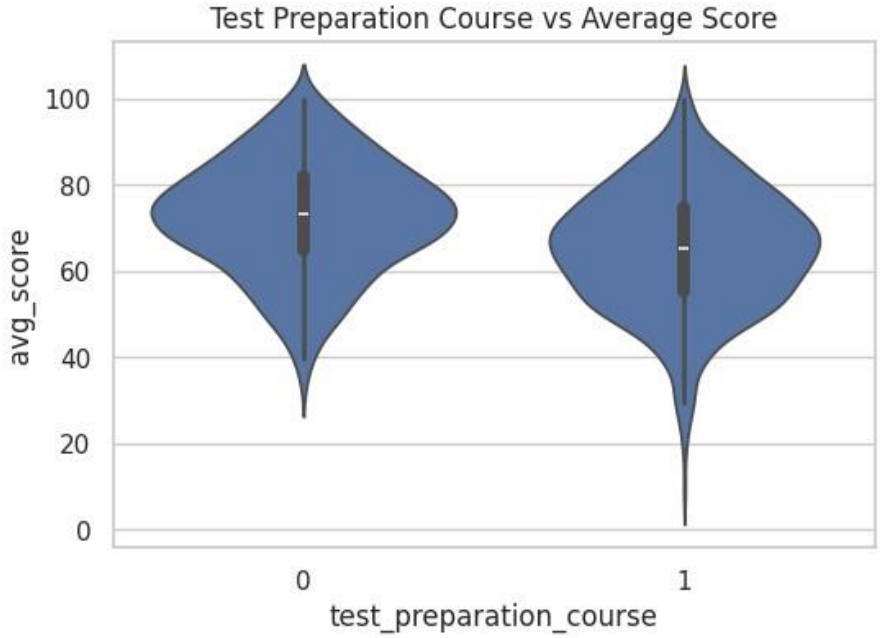
```
data=df) plt.title("Gender vs Average Score")
plt.show()
# 2. Correlation Heatmap plt.figure(figsize=(10,6))
sns.heatmap(df.corr(), annot=True,
cmap='coolwarm') plt.title("Correlation Heatmap")
plt.show()
# 3. Distribution of Scores
df[['math_score','reading_score','writing_score']].hist(figsize=(12,4)
) plt.suptitle("Distribution of Scores") plt.show()
# 4. Race/Ethnicity vs Average Score plt.figure(figsize=(6,4))
sns.barplot(x='race/ethnicity', y='avg_score', data=df,
estimator=np.mean) plt.title("Race/Ethnicity vs Average Score")
plt.show()
# 5. Test Preparation Course Impact plt.figure(figsize=(6,4))
sns.violinplot(x='test_preparation_course', y='avg_score',
data=df) plt.title("Test Preparation Course vs Average Score")
plt.show()
```


Gender vs Average Score
avg_score
100
80
60
40
20
0
1
gender


Correlation Heatmap
gender 1 -0.0015 0.0019 0.021 -0.006 0.17 -0.24 -0.3 -0.13 -0.1
race/ethnicity -0.0015 1 -0.032 0.047 -0.018 0.22 0.15 0.17 0.19 0.14
parental_level_of_education 0.0019 -0.032 1 0.0063 -0.024 -0.068 -0.072 -0.084 -0.079 -0.042
lunch 0.021 0.047 0.0063 1 0.017 0.35 0.23 0.25 0.29 0.25
test_preparation_course -0.006 -0.018 -0.024 0.017 1 -0.18 -0.24 -0.31 -0.26 -0.19
math_score 0.17 0.22 -0.068 0.35 -0.18 1 0.82 0.8 0.92 0.7
reading_score -0.24 0.15 -0.072 0.23 -0.24 0.82 1 0.95 0.97 0.75
writing_score -0.3 0.17 -0.084 0.25 -0.31 0.8 0.95 1 0.97 0.76
avg_score -0.13 0.19 -0.079 0.29 -0.26 0.92 0.97 0.97 1 0.77
result -0.1 0.14 -0.042 0.25 -0.19 0.7 0.75 0.76 0.77 1
gender
race/ethnicity
parental_level_of_education
lunch
test_preparation_course
math_score
reading_score
writing_score
avg_score
result
1.0
0.8
0.6
0.4
0.2
0.0
−0.2


Distribution of Scores
math_score
reading_score
writing_score
0
100
200
0
20
40
60
80
100


Race/Ethnicity vs Average Score
avg_score
race/ethnicity
0
10
20
30
40
50
60
70
0
1
2
3
4


Test Preparation Course vs Average Score
avg_score
test_preparation_course
0
20
40
60
80
100
0
1

```
# Step 5: Model Building (Deep Learning - MLP)
model = Sequential([
    Dense(64, activation='relu', input_shape=(X_train.shape[1],)),
    Dropout(0.3),
    Dense(32, activation='relu'),
    Dropout(0.2),
    Dense(1, activation='sigmoid')
])
model.compile(optimizer=Adam(learning_rate=0.001
),              loss='binary_crossentropy',
metrics=['accuracy']) print("\nModel Summary:")
model.summary()
```

```
Model Summary:
Model: "sequential_1"
```

Layer (type)	Output Shape	Param #
dense_3 (Dense)	(None, 64)	576
dropout_2 (Dropout)	(None, 64)	0
dense_4 (Dense)	(None, 32)	2,080
dropout_3 (Dropout)	(None, 32)	0
dense_5 (Dense)	(None, 1)	33

```
Total params: 2,689 (10.50 KB)
Trainable params: 2,689 (10.50 KB)
Non-trainable params: 0 (0.00 B)
```

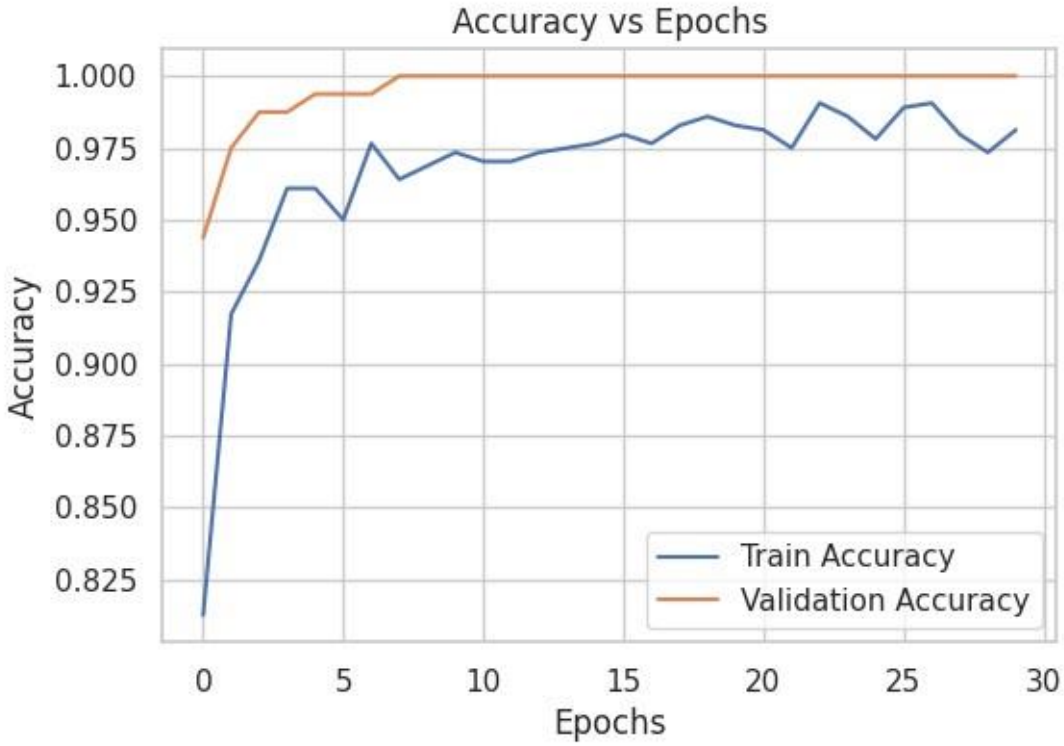
```
# Step 6: Model Training
history = model.fit(X_train, y_train, validation_split=0.2,
epochs=30, batch_size=16, verbose=1)
```

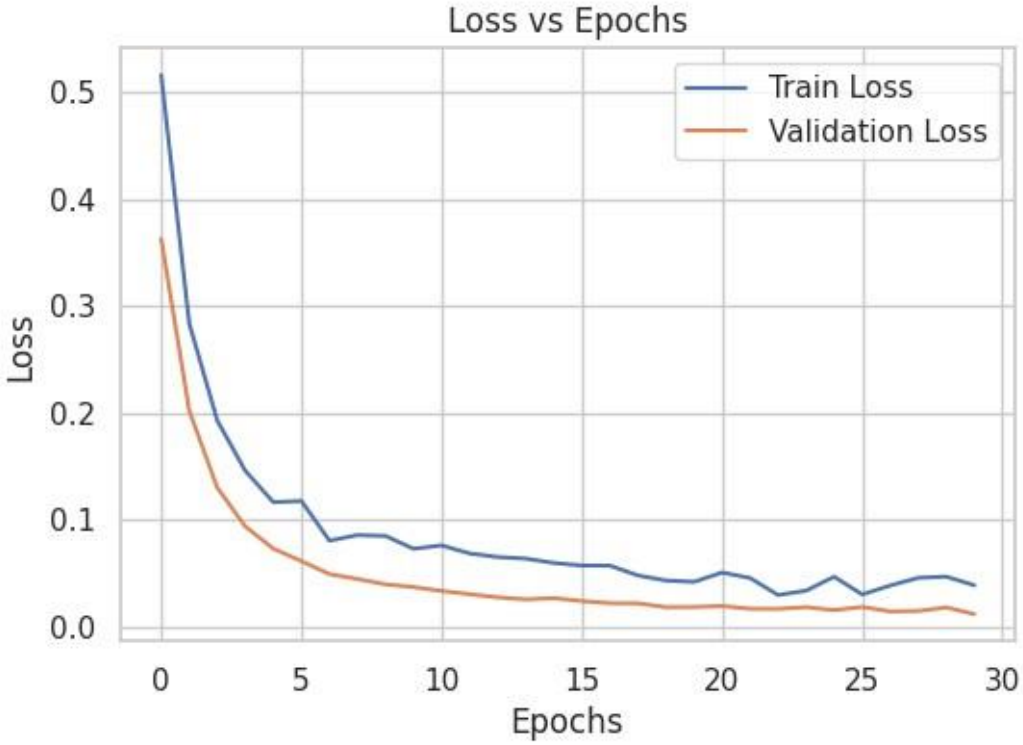
```
Epoch 1/30
40/40 ━━━━━━━━━━━━━━━━━━━━ 1s 7ms/step - accuracy: 0.6908 - loss: 0.6134 - val_accuracy: 0.9438 - val_loss: 0.3629
Epoch 2/30
40/40 ━━━━━━━━━━━━━━━━━━━━ 0s 4ms/step - accuracy: 0.9120 - loss: 0.3063 - val_accuracy: 0.9750 - val_loss: 0.2020
Epoch 3/30
40/40 ━━━━━━━━━━━━━━━━━━━━ 0s 3ms/step - accuracy: 0.9236 - loss: 0.2213 - val_accuracy: 0.9875 - val_loss: 0.1299
Epoch 4/30
40/40 ━━━━━━━━━━━━━━━━━━━━ 0s 3ms/step - accuracy: 0.9572 - loss: 0.1540 - val_accuracy: 0.9875 - val_loss: 0.0936
Epoch 5/30
40/40 ━━━━━━━━━━━━━━━━━━━━ 0s 3ms/step - accuracy: 0.9735 - loss: 0.1102 - val_accuracy: 0.9937 - val_loss: 0.0729
Epoch 6/30
40/40 ━━━━━━━━━━━━━━━━━━━━ 0s 3ms/step - accuracy: 0.9567 - loss: 0.1085 - val_accuracy: 0.9937 - val_loss: 0.0612
Epoch 7/30
40/40 ━━━━━━━━━━━━━━━━━━━━ 0s 4ms/step - accuracy: 0.9717 - loss: 0.0847 - val_accuracy: 0.9937 - val_loss: 0.0491
Epoch 8/30
40/40 ━━━━━━━━━━━━━━━━━━━━ 0s 4ms/step - accuracy: 0.9624 - loss: 0.0825 - val_accuracy: 1.0000 - val_loss: 0.0445
Epoch 9/30
40/40 ━━━━━━━━━━━━━━━━━━━━ 0s 4ms/step - accuracy: 0.9682 - loss: 0.0781 - val_accuracy: 1.0000 - val_loss: 0.0395
Epoch 10/30
40/40 ━━━━━━━━━━━━━━━━━━━━ 0s 3ms/step - accuracy: 0.9657 - loss: 0.0787 - val_accuracy: 1.0000 - val_loss: 0.0369
Epoch 11/30
40/40 ━━━━━━━━━━━━━━━━━━━━ 0s 4ms/step - accuracy: 0.9678 - loss: 0.0854 - val_accuracy: 1.0000 - val_loss: 0.0333
Epoch 12/30
40/40 ━━━━━━━━━━━━━━━━━━━━ 0s 3ms/step - accuracy: 0.9718 - loss: 0.0713 - val_accuracy: 1.0000 - val_loss: 0.0302
Epoch 13/30
40/40 ━━━━━━━━━━━━━━━━━━━━ 0s 3ms/step - accuracy: 0.9819 - loss: 0.0535 - val_accuracy: 1.0000 - val_loss: 0.0274
Epoch 14/30
40/40 ━━━━━━━━━━━━━━━━━━━━ 0s 4ms/step - accuracy: 0.9852 - loss: 0.0508 - val_accuracy: 1.0000 - val_loss: 0.0254
Epoch 15/30
40/40 ━━━━━━━━━━━━━━━━━━━━ 0s 3ms/step - accuracy: 0.9782 - loss: 0.0620 - val_accuracy: 1.0000 - val_loss: 0.0266
Epoch 16/30
40/40 ━━━━━━━━━━━━━━━━━━━━ 0s 3ms/step - accuracy: 0.9835 - loss: 0.0481 - val_accuracy: 1.0000 - val_loss: 0.0238
```

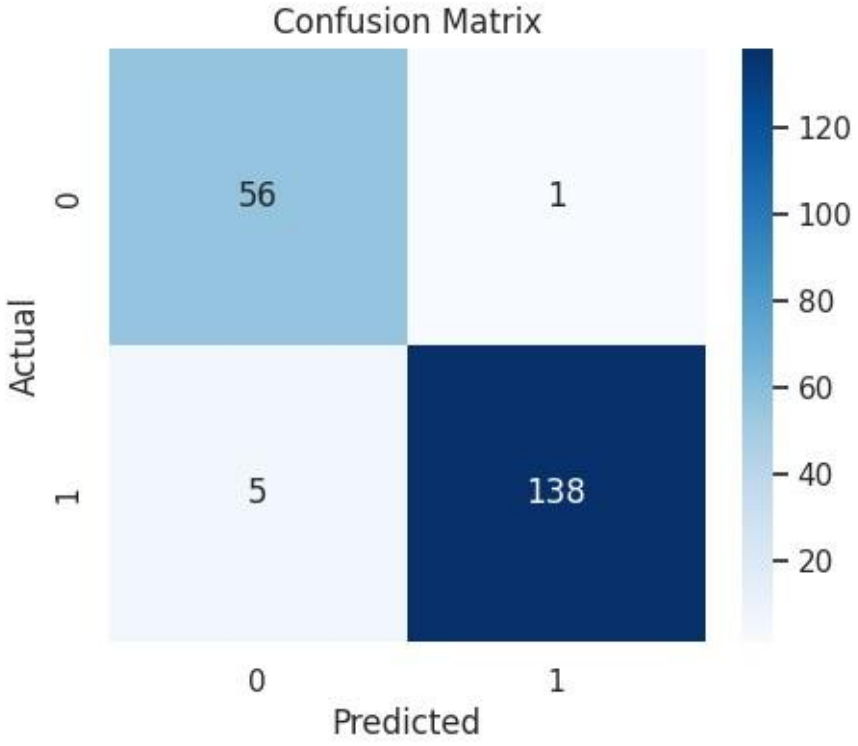
```
Epoch 17/30
40/40 ━━━━━━━━━━━━━━━━━━━━ 0s 3ms/step - accuracy: 0.9835 - loss: 0.0458 - val_accuracy: 1.0000 - val_loss: 0.0219
Epoch 18/30
40/40 ━━━━━━━━━━━━━━━━━━━━ 0s 3ms/step - accuracy: 0.9908 - loss: 0.0343 - val_accuracy: 1.0000 - val_loss: 0.0218
Epoch 19/30
40/40 ━━━━━━━━━━━━━━━━━━━━ 0s 3ms/step - accuracy: 0.9823 - loss: 0.0477 - val_accuracy: 1.0000 - val_loss: 0.0181
Epoch 20/30
40/40 ━━━━━━━━━━━━━━━━━━━━ 0s 4ms/step - accuracy: 0.9754 - loss: 0.0514 - val_accuracy: 1.0000 - val_loss: 0.0182
Epoch 21/30
40/40 ━━━━━━━━━━━━━━━━━━━━ 0s 4ms/step - accuracy: 0.9851 - loss: 0.0462 - val_accuracy: 1.0000 - val_loss: 0.0192
Epoch 22/30
40/40 ━━━━━━━━━━━━━━━━━━━━ 0s 3ms/step - accuracy: 0.9843 - loss: 0.0370 - val_accuracy: 1.0000 - val_loss: 0.0167
Epoch 23/30
40/40 ━━━━━━━━━━━━━━━━━━━━ 0s 4ms/step - accuracy: 0.9945 - loss: 0.0291 - val_accuracy: 1.0000 - val_loss: 0.0165
Epoch 24/30
40/40 ━━━━━━━━━━━━━━━━━━━━ 0s 3ms/step - accuracy: 0.9903 - loss: 0.0270 - val_accuracy: 1.0000 - val_loss: 0.0182
Epoch 25/30
40/40 ━━━━━━━━━━━━━━━━━━━━ 0s 3ms/step - accuracy: 0.9726 - loss: 0.0450 - val_accuracy: 1.0000 - val_loss: 0.0154
Epoch 26/30
40/40 ━━━━━━━━━━━━━━━━━━━━ 0s 4ms/step - accuracy: 0.9921 - loss: 0.0271 - val_accuracy: 1.0000 - val_loss: 0.0183
Epoch 27/30
40/40 ━━━━━━━━━━━━━━━━━━━━ 0s 4ms/step - accuracy: 0.9926 - loss: 0.0402 - val_accuracy: 1.0000 - val_loss: 0.0142
Epoch 28/30
40/40 ━━━━━━━━━━━━━━━━━━━━ 0s 3ms/step - accuracy: 0.9814 - loss: 0.0470 - val_accuracy: 1.0000 - val_loss: 0.0146
Epoch 29/30
40/40 ━━━━━━━━━━━━━━━━━━━━ 0s 3ms/step - accuracy: 0.9866 - loss: 0.0304 - val_accuracy: 1.0000 - val_loss: 0.0179
Epoch 30/30
40/40 ━━━━━━━━━━━━━━━━━━━━ 0s 3ms/step - accuracy: 0.9845 - loss: 0.0311 - val_accuracy: 1.0000 - val_loss: 0.0116
```

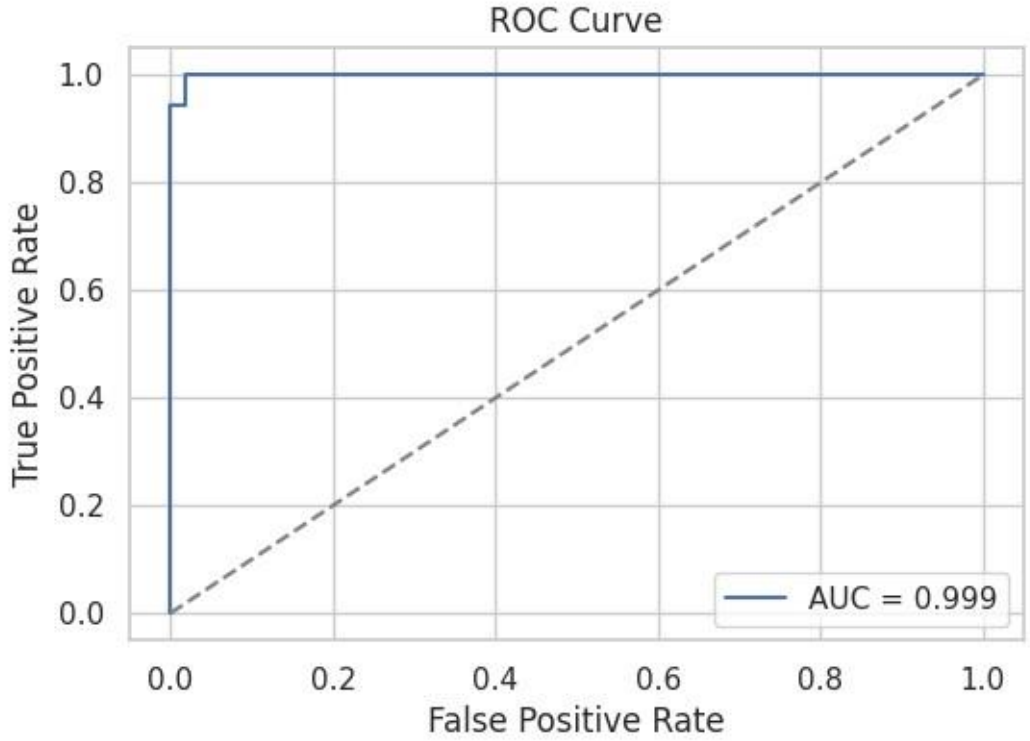
```
# Step 7: Model Evaluation Visualization
# Accuracy vs Epochs
plt.figure(figsize=(6,4))
plt.plot(history.history['accuracy'], label='Train Accuracy')
plt.plot(history.history['val_accuracy'], label='Validation Accuracy') plt.xlabel("Epochs")
plt.ylabel("Accuracy") plt.title("Accuracy vs Epochs") plt.legend() plt.show() # Loss vs
```

```
Epochs plt.figure(figsize=(6,4)) plt.plot(history.history['loss'], label='Train Loss')
plt.plot(history.history['val_loss'], label='Validation Loss') plt.xlabel("Epochs")
plt.ylabel("Loss") plt.title("Loss vs Epochs") plt.legend() plt.show() # Predictions
y_pred_prob = model.predict(X_test) y_pred = (y_pred_prob >= 0.5).astype(int)
# Confusion Matrix cm = confusion_matrix(y_test,
y_pred) plt.figure(figsize=(5,4)) sns.heatmap(cm,
annot=True, fmt="d", cmap="Blues")
plt.xlabel("Predicted") plt.ylabel("Actual")
plt.title("Confusion Matrix") plt.show() # ROC Curve
fpr, tpr, _ = roc_curve(y_test, y_pred_prob)
roc_auc = auc(fpr, tpr) plt.figure(figsize=(6,4))
plt.plot(fpr, tpr, label=f"AUC =
{roc_auc:.3f}") plt.plot([0,1],[0,1],'--
',color='gray') plt.xlabel("False Positive Rate")
plt.ylabel("True Positive Rate") plt.title("ROC
Curve")
plt.legend() plt.show()
print("\nClassification Report:")
print(classification_report(y_test,
y_pred))
```


Accuracy vs Epochs
1.000
0.975
0.950
0.925
0.900
0.875
0.850
0.825
Accuracy
0
5
10
15
20
25
30
Epochs
Train Accuracy
Validation Accuracy


Loss vs Epochs
0.5
0.4
0.3
0.2
0.1
0.0
Loss
0
5
10
15
20
25
30
Epochs
Train Loss
Validation Loss


Confusion Matrix
56
1
5
138
Actual
0
1
Predicted
0
1
120
100
80
60
40
20


ROC Curve
True Positive Rate
False Positive Rate
0.0
0.2
0.4
0.6
0.8
1.0
AUC = 0.999

```
Classification Report:
              precision    recall  f1-score   support

           0       0.92      0.98      0.95        57
           1       0.99      0.97      0.98       143

    accuracy                           0.97       200
   macro avg       0.96      0.97      0.96       200
weighted avg       0.97      0.97      0.97       200
```

```
# Step 8: Conclusion & Future Scope
print("\n=== CONCLUSION ===")
print("- Students who completed the test preparation course performed better overall.")
print("- All three scores (math, reading, writing) are highly correlated.") print("- Model achieved good accuracy and AUC, showing ability to predict pass/fail effectively.")
print("- Further improvement could include more features like attendance, study time, or practical marks.") print("\n=== FUTURE SCOPE ===") print("- Try CNN/LSTM models if sequential/temporal features are available.") print("- Hyperparameter tuning with KerasTuner or GridSearchCV could improve results.") print("- Apply explainable AI (SHAP/LIME) for feature importance insights.")
```

```
=== CONCLUSION ===
- Students who completed the test preparation course performed better overall.
- All three scores (math, reading, writing) are highly correlated.
- Model achieved good accuracy and AUC, showing ability to predict pass/fail effectively.
- Further improvement could include more features like attendance, study time, or practical marks.

=== FUTURE SCOPE ===
- Try CNN/LSTM models if sequential/temporal features are available.
- Hyperparameter tuning with KerasTuner or GridSearchCV could improve results.
- Apply explainable AI (SHAP/LIME) for feature importance insights.
```