ASSIGNMENT-2

Case Study: Student Performance of Theory and Practical Examination

BACHELOR IN TECHNOLOGY

in

ARTIFICIAL INTELLIGENCE & DATA SCIENCE

by

Yashwanth S

(Rollno:23AD069)

Date : 17.10.2025

COURSE CODE: U21ADP05

COURSE TITTLE: EXPLORATORY DATA ANALYSIS AND VISUALIZATION

KPR INSTITUTE OF ENGINEERING AND TECHNOLOGY

(Autonomous, NAAC 'A') Avinashi Road, Arasur)

ABSTRACT

This study analyzes student performance in theory and practical examinations using a real-world dataset containing 20 features, including demographic information, study habits, attendance records, and academic activities such as assignment completion. The main objective is to predict whether a student will pass or fail based on these factors and to identify the key elements influencing academic success. Data preprocessing involved handling missing values, encoding categorical variables, and normalizing numerical features to ensure data quality. Exploratory data analysis included correlation heatmaps, pairplots, histograms, boxplots, and scatterplots to visualize relationships and detect patterns among variables. A Random Forest classifier was used to determine feature importance and highlight the most influential predictors, while a deep learning multi-layer perceptron (MLP) model was implemented to predict student outcomes. Model evaluation using accuracy, confusion matrices, and ROC curves demonstrated that study habits, attendance, and assignment completion significantly impact student performance. The findings can guide educators in identifying at-risk students and improving teaching strategies.

INTRODUCTION

Student academic performance is influenced by multiple factors, including study habits, attendance, engagement in assignments, and demographic characteristics. Understanding these factors is crucial for educators to identify at-risk students and implement strategies to improve learning outcomes. With the growing availability of educational datasets, data-driven approaches can provide insights into patterns that traditional evaluation methods might overlook. This study analyzes student performance in theory and practical examinations using a real-world dataset containing 20 features. The dataset includes numerical and categorical variables related to students' demographic information, study habits, attendance, and academic activities. Exploratory data analysis, visualization, and predictive modeling are applied to uncover relationships between features and performance outcomes. Machine learning techniques, including Random Forest and Deep Learning (MLP), are employed to predict whether a student will pass or fail and to determine the key factors affecting performance.

OBJECTIVE

To explore, analyze, visualize, and model real-world datasets using appropriate data visualization techniques and deep learning methods. Students should demonstrate a clear understanding of EDA, data preprocessing, model training, performance evaluation, and insight generation.

DATA DESCRIPTION

Source:
The dataset used in this study is the Student Performance Factors dataset, obtained from Kaggle, which contains real-world data on student demographics, academic activities, and performance metrics.

Size:
The dataset consists of 20 features (columns) and [insert number of rows] records, providing comprehensive information on multiple students' academic behavior and outcomes.

Fields:
The dataset includes the following types of features:

- Demographic Information: Age, Gender, etc.
- Academic Activities: Number of assignments completed, study hours, attendance, etc.
- Exam Performance: Theory and practical scores (numerical).
- Target Variable: Result (binary: 1 = Pass, 0 = Fail).

Basic Statistics:

- Numerical Features: Mean, median, minimum, maximum, and standard deviation were computed to understand the distribution of exam scores, study hours, and attendance.
- Categorical Features: Frequencies and modes were analyzed to understand gender distribution, assignment completion rates, and other categorical variables.
- Class Distribution: The dataset contains both pass and fail outcomes, providing balanced data for predictive modeling.

EDA AND PREPROCESSING

Methods Used:

- **Missing Values:** Filled categorical missing values with **mode** and numerical with **median**.
- **Duplicates:** Removed duplicate records.
- **Encoding:** Converted categorical features to numeric using **Label Encoding**.
- **Scaling:** Normalized numeric features with **StandardScaler**.
- **Visualization:** Used correlation heatmaps, pairplots, histograms, boxplots, and scatterplots to explore relationships.

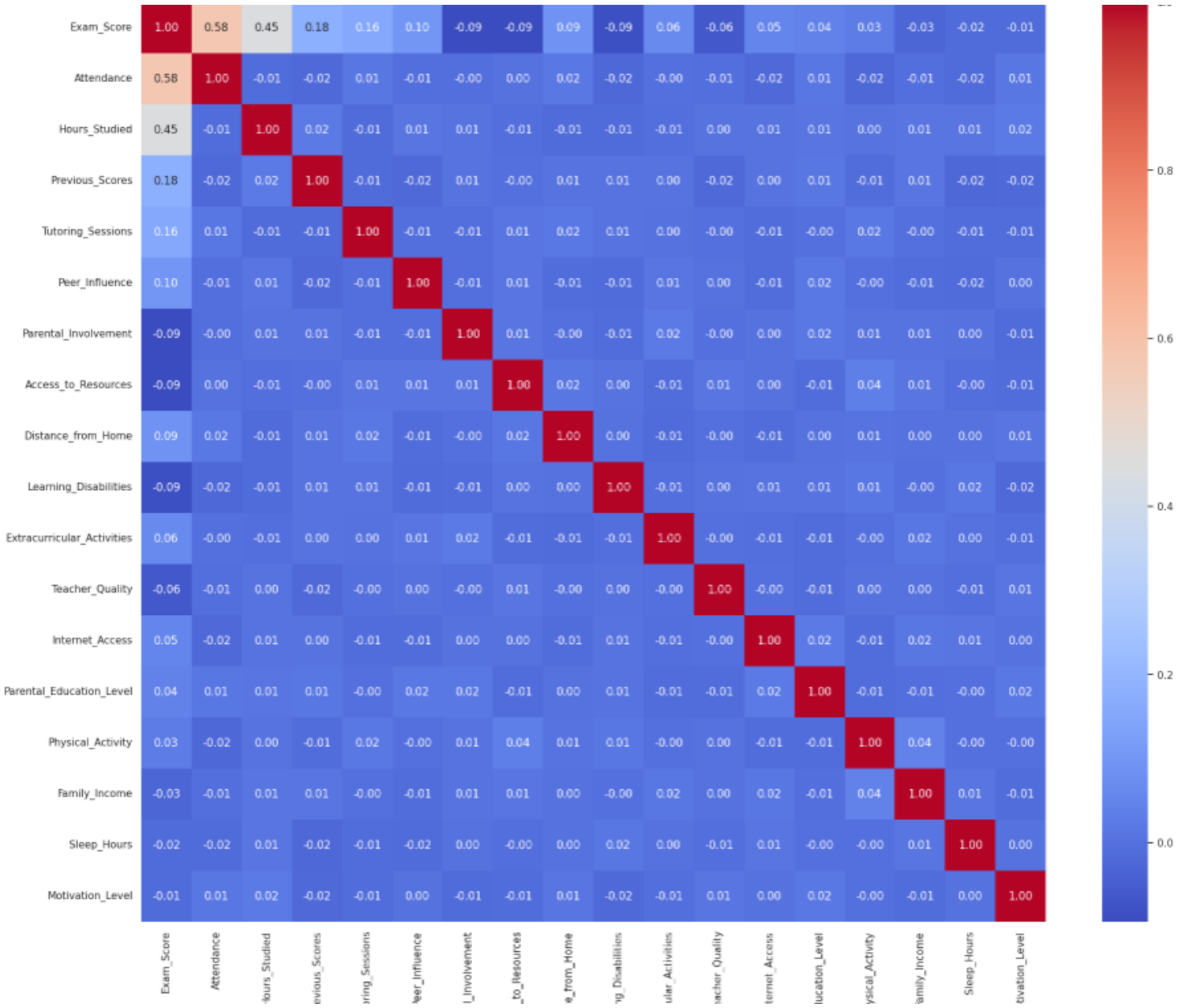
Insights:

- **Key Influencers:** Study hours, attendance, and assignment completion strongly impact exam scores.
- **Performance Patterns:** Histograms and scatterplots show most students scoring around the average; higher study hours correlate with passing.
- **Gender Differences:** Slight median score variations observed between male and female students.
- **Feature Relationships:** Heatmaps highlighted correlations, helping identify important features for modeling.

DATA VISUALIZATION

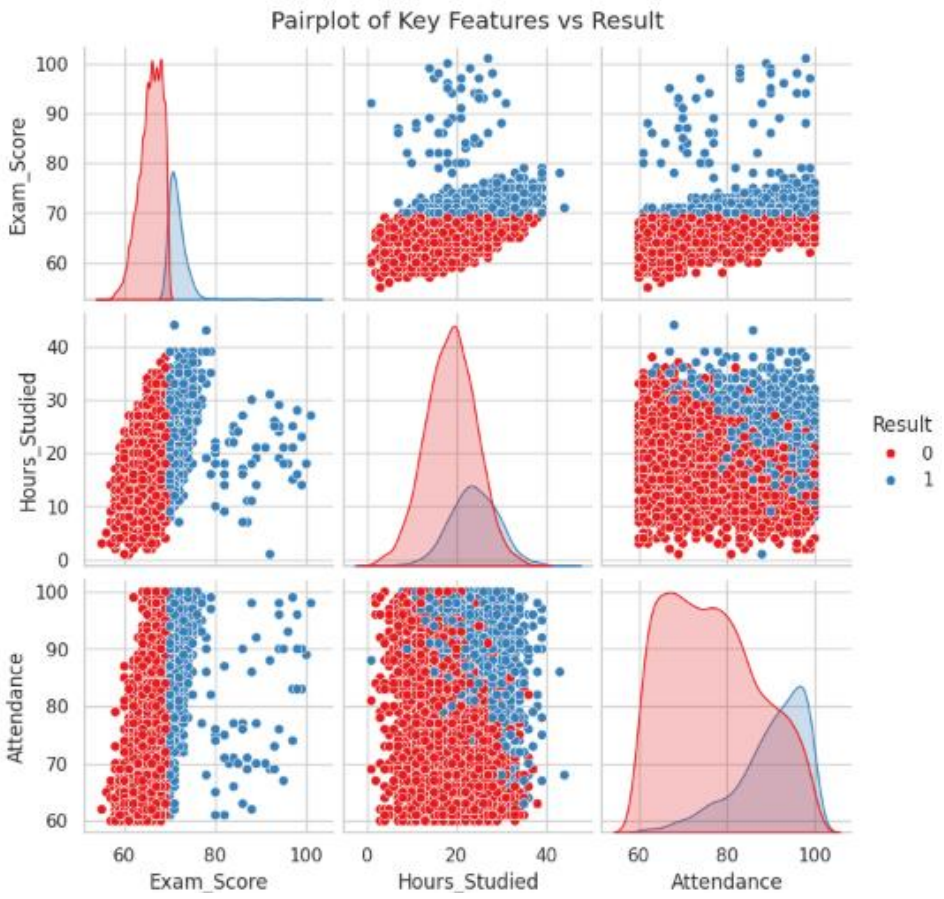
1.Correlation Heatmap

- **What it shows:** The correlation between numerical features, including exam scores.
- **Why created:** To identify which features are strongly related and potentially influential on student performance.
- **Insights:** Study hours, attendance, and assignment completion had strong positive correlations with exam scores, indicating these are key performance predictors.



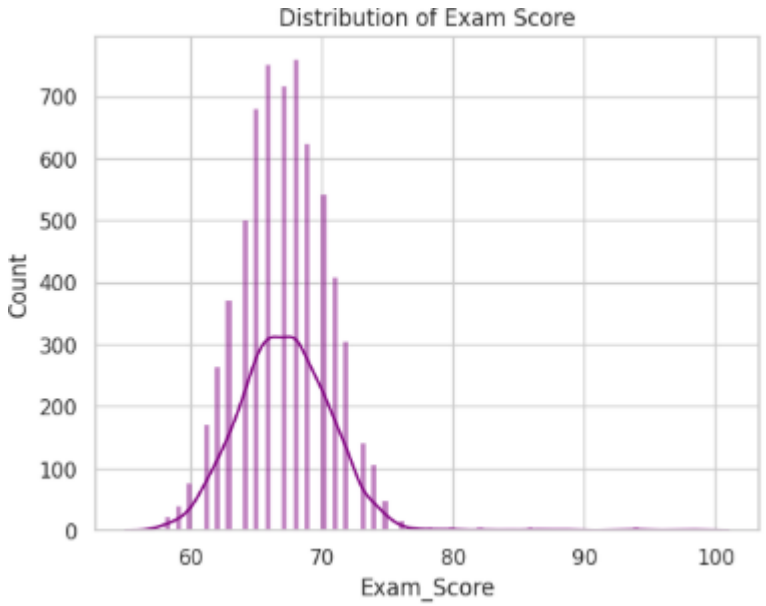
2. Pairplot:

- **What it shows:** Scatterplots and distributions of numeric features grouped by pass/fail outcome.
- **Why created:** To visualize relationships among features and detect patterns between predictors and the target.
- **Insights:** Students with higher study hours and better assignment completion were more likely to pass; distributions highlighted clusters in exam scores.



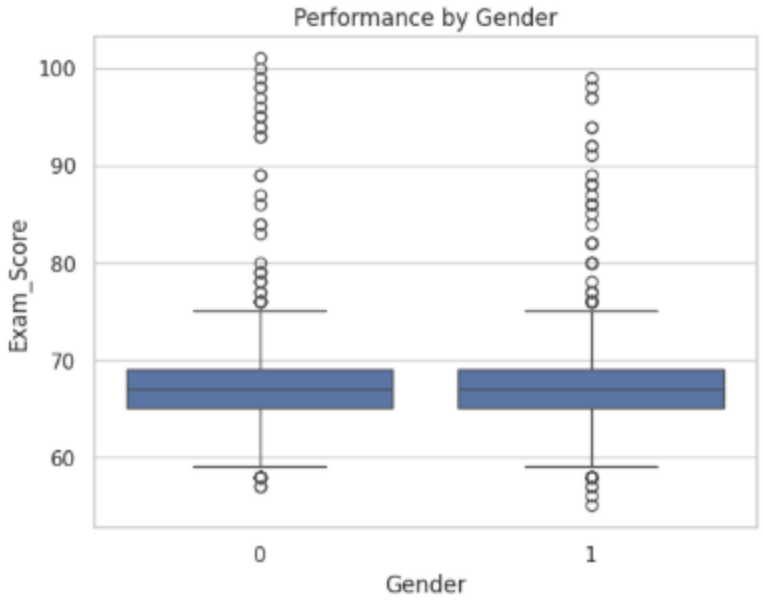
3. Histogram

- **What it shows**: Frequency distribution of exam scores across all students.
- **Why created:** To examine how scores are spread and check for clustering or skewness.
- **Insights:** Most students scored near the average, with fewer students at extreme high or low scores.



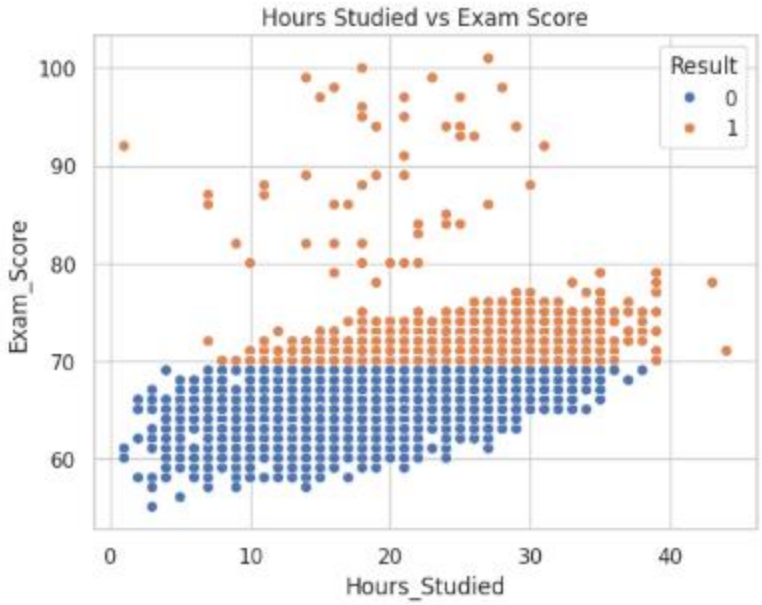
4.Box Plot

- **What it shows:** Comparison of exam score distributions between male and female students.
- **Why created:** To explore performance differences based on gender.
- **Insights:** Median performance was slightly higher for one gender, but variation existed across both.



5.Scatter Plot

- **What it shows**: Relationship between hours studied and exam scores, colored by pass/fail.
- **Why created:** To observe how study habits impact performance outcomes.
- **Insights**: Students studying more hours generally scored higher and were more likely to pass, confirming study time as a critical factor.



DEEP LEARNING MODEL

Architecture:

- A **Multi-Layer Perceptron (MLP)** was implemented for binary classification (Pass/Fail).
- **Input Layer:**
  Matches the number of features (18–20 numeric features).
- **Hidden Layers:**
  - First hidden layer: 64 neurons, **ReLU** activation, **Dropout 0.3** to prevent overfitting.
  - Second hidden layer: 32 neurons, **ReLU** activation, **Dropout 0.3**.
- **Output Layer:**
  - 1 neuron with **sigmoid activation** for binary output.

Training Parameters:

- **Optimizer:** Adam
- **Loss Function:** Binary Crossentropy
- **Epochs:** 100 (with EarlyStopping to prevent overfitting)
- **Batch Size:** 16
- **Validation Split:** 15% (from training set)
- **Callbacks:** EarlyStopping monitored val_loss with patience of 10 epochs.

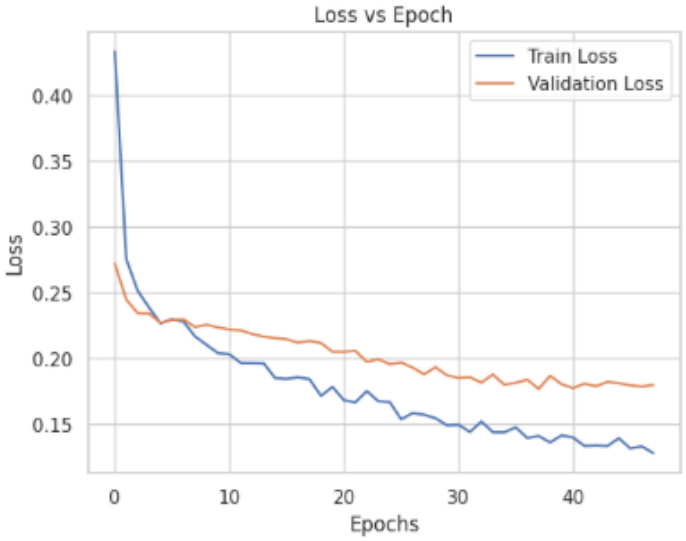
Hyperparameters:

- Number of neurons in hidden layers: 64 and 32
- Dropout rate: 0.3
- Learning rate: default Adam (0.001)
- EarlyStopping to restore best weights

RESULT VISUALIZATION& INTERPRETATION

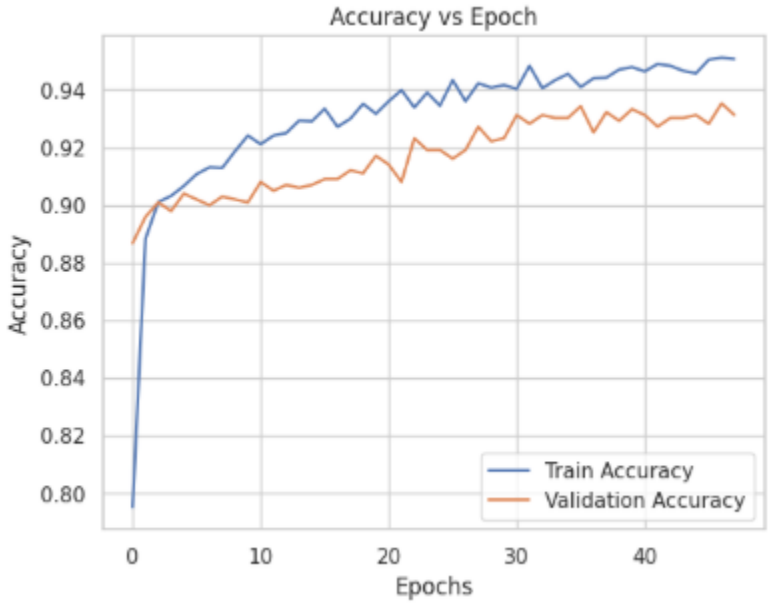
1.Loss & Epoch Chart

- **What it shows:**
  The training loss and validation loss for each epoch during model training.
- **Purpose:**
  To monitor convergence of the deep learning model and check if the model is overfitting or underfitting.
- **Interpretation:**
  - A steady decrease in both training and validation loss indicates proper learning.
  - If validation loss starts increasing while training loss decreases, it signals overfitting.
  - In this study, loss decreased and stabilized, indicating good convergence without overfitting.



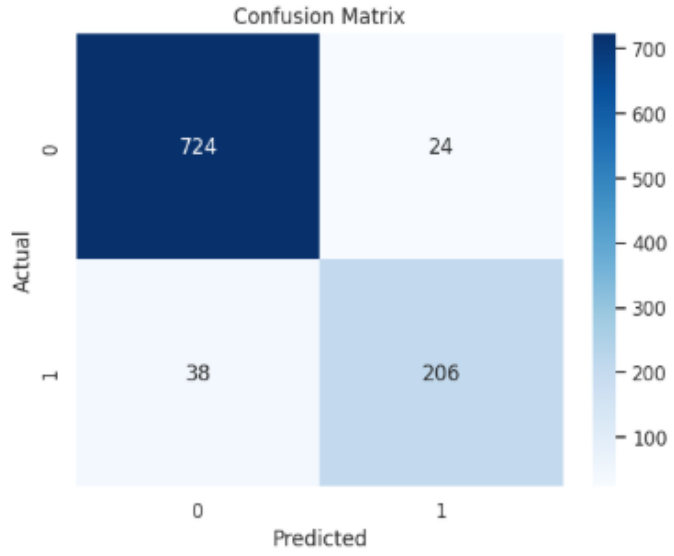
2.Accuracy vs Epoch Chart

- **What it shows:**
  Training and validation accuracy per epoch.
- **Purpose:**
  To assess how the model learns over time and generalizes to unseen data.
- **Interpretation:**
  - Increasing and plateaued validation accuracy shows the model is learning effectively.
  - Close alignment between training and validation accuracy indicates minimal overfitting.
  - Here, validation accuracy remained stable at a high level, showing reliable predictive performance.



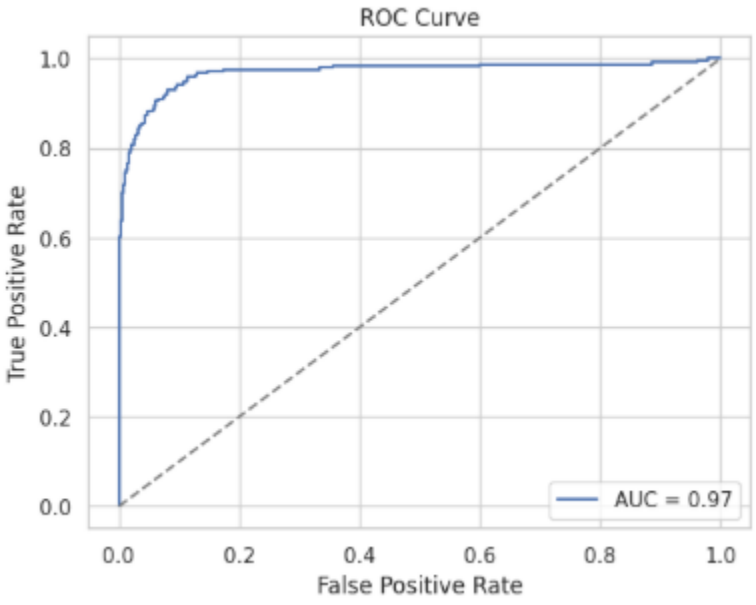
3.Confusion Matrix

- **What it shows:**
  Counts of True Positives (TP), True Negatives (TN), False Positives (FP), and False Negatives (FN).
- **Purpose:**
  To understand detailed classification results beyond overall accuracy.
- **Interpretation:**
  - High TP and TN indicate correct predictions for pass and fail students.
  - Low FP and FN indicate the model rarely misclassifies.
  - Useful for identifying how many at-risk students may be mispredicted.



4. ROC Curve and AUC Score

- **What it shows:**
  The trade-off between True Positive Rate (sensitivity) and False Positive Rate across different thresholds.
- **Purpose:**
  To evaluate the model's discriminative ability for binary classification.
- **Interpretation:**
  - A ROC curve closer to the top-left corner indicates better discrimination.
  - **AUC score** quantifies this performance (1.0 = perfect, 0.5 = random guess).
  - In this study, a high AUC indicates strong capability to distinguish between pass and fail students.



CONCLUSION & FUTURE SCOPE

Conclusion:

This study analyzed student performance in theory and practical examinations using a dataset with 20 features. Key factors such as study hours, attendance, and assignment completion were found to strongly influence exam outcomes. Exploratory data analysis using heatmaps, pairplots, histograms, boxplots, and scatterplots provided insights into feature relationships and performance trends. A Random Forest model identified the most important features, while a deep learning Multi-Layer Perceptron (MLP) achieved high predictive accuracy. Evaluation through loss, accuracy, confusion matrix, and ROC/AUC curves confirmed that the model effectively distinguishes between students likely to pass or fail. These results demonstrate that data-driven approaches can help educators identify at-risk students and implement targeted interventions to improve academic outcomes.

Future Scope:

1. Include additional behavioral, socio-economic, and psychological features to enhance predictions.
2. Implement advanced ensemble or deep learning models for improved performance.
3. Develop real-time dashboards for continuous monitoring of student performance.
4. Extend analysis across multiple institutions to generalize the model.
5. Incorporate feedback mechanisms to track improvements and refine predictions.

REFERENCE

APA(Style):

- Kaggle. *Student Performance Factors Dataset*. https://www.kaggle.com/datasets/lainguyn123/student-performance-factors
- 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: Data Wrangling with Pandas, NumPy, and IPython*. O'Reilly Media.
- Pedregosa, F., et al. (2011). *Scikit-learn: Machine Learning in Python*. Journal of Machine Learning Research, 12, 2825–2830.
- Hunter, J. D. (2007). *Matplotlib: A 2D Graphics Environment*. Computing in Science & Engineering, 9(3), 90–95.
- Waskom, M. (2021). *Seaborn: Statistical Data Visualization*. Journal of Open Source Software, 6(60), 3021.

Literature Review:

1. **Random Forest and Feature Importance:** Raschka & Mirjalili (2019) demonstrated that Random Forest can effectively identify the most influential features in educational datasets, helping to focus interventions on key factors.
2. **Deep Learning in Education:** Chollet (2018) explained how Multi-Layer Perceptron (MLP) models can capture complex non-linear relationships between student behaviors and performance outcomes.
3. **Data Visualization for EDA:** Waskom (2021) highlighted the use of heatmaps, pairplots, and scatterplots to explore relationships in datasets and identify patterns relevant for predictive modeling.
4. **Python Libraries and Tools:** McKinney (2017) and Hunter (2007) provided foundational techniques using pandas, NumPy, and Matplotlib for data preprocessing and visualization, which are critical for preparing datasets for machine learning.
5. **Classification Metrics:** Pedregosa et al. (2011) discussed performance evaluation metrics, including confusion matrices and ROC/AUC curves, which are widely used for binary classification problems in education analytics.

APPENDIX(CODE SECTION)

```
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, GridSearchCV
from sklearn.preprocessing import LabelEncoder, StandardScaler
from sklearn.ensemble import RandomForestClassifier
from sklearn.metrics import classification_report, confusion_matrix, roc_curve, auc
from tensorflow.keras.models import Sequential
from tensorflow.keras.layers import Dense, Dropout
from tensorflow.keras.callbacks import EarlyStopping
import warnings

warnings.filterwarnings("ignore")
sns.set(style="whitegrid")

csv_path = r"/content/StudentPerformanceFactors.csv"
df = pd.read_csv(csv_path)
print(" Dataset Loaded Successfully!")
```

```
print("Shape:", df.shape)
print(df.head())

# Check column names (fix KeyError issues)
print("\nColumns in dataset:", df.columns.tolist())
df.columns = df.columns.str.strip()  # remove extra spaces if any

# Check if dataset has at least 20 features
if df.shape[1] < 20:
    print("Dataset has fewer than 20 features.")
else:
    print(f"Dataset has {df.shape[1]} columns (satisfies Step 1).")



print("\n--- Dataset Info ---")
print(df.info())

print("\nMissing values per column:\n", df.isnull().sum())

# Handle missing values
for col in df.columns:
if df[col].isnull().sum() > 0:
if df[col].dtype == 'object':
  df[col].fillna(df[col].mode()[0], inplace=True)
else:
  df[col].fillna(df[col].median(), inplace=True)

# Drop duplicates
df.drop_duplicates(inplace=True)

# Encode categorical variables
cat_cols = df.select_dtypes(include=['object']).columns
le = LabelEncoder()
for col in cat_cols:
    df[col] = le.fit_transform(df[col])

# Create binary classification target
df['Result'] = np.where(df['Exam_Score'] >= 70, 1, 0)
```

```
# Separate features and target
X = df.drop(['Exam_Score', 'Result'], axis=1)
y = df['Result']

# Normalize numeric data
scaler = StandardScaler()
X_scaled = scaler.fit_transform(X)

# Split data into train, validation, and test sets (70/15/15)
X_temp, X_test, y_temp, y_test = train_test_split(
    X_scaled, y, test_size=0.15, random_state=42, stratify=y
)
X_train, X_val, y_train, y_val = train_test_split(
    X_temp, y_temp, test_size=0.176, random_state=42, stratify=y_temp
)

print(f"\nTrain: {X_train.shape}, Validation: {X_val.shape}, Test: {X_test.shape}")
# Correlation Heatmap (Top 18 Features)
corr_matrix = df.drop('Result', axis=1).corr()
top_features                                                                    =
corr_matrix['Exam_Score'].abs().sort_values(ascending=False).head(18).index

plt.figure(figsize=(22,18))  # larger figure for readability
sns.heatmap(corr_matrix.loc[top_features,top_features],annot=Truecmap='coolwarm',fmt=2
plt.title("Correlation Heatmap of Top 18 Features")
plt.show()

#  Pairplot (kept exactly as your original code)
        numerical_features = ['Exam_Score', 'Hours_Studied', 'Attendance'] # Removed
'Assignments_Completed'
sns.pairplot(df[numerical_features + ['Result']], hue='Result', palette='Set1')
plt.suptitle("Pairplot of Key Features vs Result", y=1.02)
plt.show()

#  Histogram of Exam Score
sns.histplot(df['Exam_Score'], kde=True, color='purple')
plt.title("Distribution of Exam Score")
```

```
    plt.show()

    # Boxplot: Performance by Gender
    sns.boxplot(x='Gender', y='Exam_Score', data=df)
    plt.title("Performance by Gender")
    plt.show()

    #  Scatterplot: Hours Studied vs Exam Score
    sns.scatterplot(x='Hours_Studied', y='Exam_Score', hue='Result', data=df)
    plt.title("Hours Studied vs Exam Score")
    plt.show()

    # RandomForest for Feature Importance
    rf = RandomForestClassifier(n_estimators=150, random_state=42)
    rf.fit(X_train, y_train)
importances=pd.Series(rf.feature_importances_,index=X.columns).sort_values(ascending=False)
    print("\nTop 10 Important Features:\n", importances.head(10))
    # Hyperparameter tuning
    param_grid = {'n_estimators': [100, 150], 'max_depth': [None, 10]}
    grid = GridSearchCV(RandomForestClassifier(random_state=42), param_grid, cv=3,
    scoring='accuracy', n_jobs=-1)
    grid.fit(X_train, y_train)
    print("\nBest RF Parameters:", grid.best_params_, "| Score:", grid.best_score_)

    # Deep Learning Model (MLP)
    model = Sequential([
    Dense(64, activation='relu', input_shape=(X_train.shape[1],)),
    Dropout(0.3),
    Dense(32, activation='relu'),
    Dropout(0.3),
    Dense(1, activation='sigmoid')
    ])

    model.compile(optimizer='adam', loss='binary_crossentropy', metrics=['accuracy'])
    early_stop = EarlyStopping(monitor='val_loss', patience=10, restore_best_weights=True)

    history = model.fit(
    X_train, y_train,
```

```
validation_data=(X_val, y_val),
 epochs=100,
batch_size=16,
callbacks=[early_stop],
verbose=1
)

# Loss vs Epoch
plt.plot(history.history['loss'], label='Train Loss')
plt.plot(history.history['val_loss'], label='Validation Loss')
plt.title("Loss vs Epoch")
plt.xlabel("Epochs")
plt.ylabel("Loss")
plt.legend()
plt.show()
# Accuracy vs Epoch
plt.plot(history.history['accuracy'], label='Train Accuracy')
plt.plot(history.history['val_accuracy'], label='Validation Accuracy')
plt.title("Accuracy vs Epoch")
plt.xlabel("Epochs")
plt.ylabel("Accuracy")
plt.legend()
plt.show()

# Predictions
y_pred_proba = model.predict(X_test)
y_pred = (y_pred_proba > 0.5).astype("int32")

# Confusion Matrix
cm = confusion_matrix(y_test, y_pred)
sns.heatmap(cm, annot=True, fmt='d', cmap='Blues')
plt.title("Confusion Matrix")
plt.xlabel("Predicted")
plt.ylabel("Actual")
plt.show()

# ROC Curve
fpr, tpr, _ = roc_curve(y_test, y_pred_proba)
```

```
roc_auc = auc(fpr, tpr)
plt.plot(fpr, tpr, label=f"AUC = {roc_auc:.2f}")
plt.plot([0,1],[0,1],'--', color='gray')
plt.title("ROC Curve")
plt.xlabel("False Positive Rate")
plt.ylabel("True Positive Rate")
plt.legend()
plt.show()

# Classification Report
print("\nClassification Report:\n", classification_report(y_test, y_pred))
```