Experiment 1

Aim

To implement and analyze Linear Regression and Logistic Regression models on real-world datasets using Python and evaluate their performance using appropriate metrics.

1. Dataset Source

This experiment utilizes a single expanded dataset to demonstrate both regression and classification tasks.

- **Dataset Name:** Expanded Student Performance Dataset
- **Source:** expanded_student_performance.csv
- **Description:** This dataset is an extended version of the standard student performance dataset, containing a larger number of records to ensure robust model training and evaluation.

2. Dataset Description

The dataset is used for two distinct machine learning tasks:

Task 1: Linear Regression (Score Prediction)

- **Type:** Numerical, Tabular
- **Features:** Hours_Studied, Attendance, Assignment_Score, Midterm_Score.
- **Target:** Final_Score (Continuous variable).
- **Goal:** Predict the exact final score a student will achieve based on their study habits and interim grades.

Task 2: Logistic Regression (Binary Classification)

- **Type:** Binary Classification
- **Features:** Hours_Studied, Attendance, Assignment_Score, Midterm_Score.
- **Target:** Pass_Status (Binary: 1 = Pass, 0 = Fail).
    - *Note: This target variable is derived from Final_Score, where a score $\ge$ 60 is labeled as 'Pass' (1) and < 60 is labeled as 'Fail' (0).*
- **Goal:** Classify whether a student will pass or fail based on the input features.

3. Mathematical Formulation

Linear Regression (for Continuous Prediction)

- **Hypothesis ($\hat{y}$):** Represents the predicted value as a linear combination of inputs.
  $\hat{y} = \beta_0 + \beta_1 x_1 + \beta_2 x_2 + \dots + \beta_n x_n$

- **Cost Function (MSE):** Measures the average squared difference between estimated and actual values.
  $MSE = (1/N) \sum (y_i - \hat{y}_i)^2$
- **Gradient Descent:** Iterative optimization to minimize the cost function.
  $\beta_{□} := \beta_{□} - \alpha (\partial J / \partial \beta_{□})$

Logistic Regression (for Binary Classification)

- **Sigmoid Function (σ):** Maps any real-valued number into the probability range [0, 1].
  $\sigma(z) = 1 / (1 + e^{-z})$
  (Where z is the linear combination of features).
- **Decision Boundary:**
  If $\sigma(z) \geq 0.5$, Predict 1 (Pass)
  If $\sigma(z) < 0.5$, Predict 0 (Fail)

4. Algorithm Limitations

- **Linearity Assumption:** Linear regression assumes a straight-line relationship between dependent and independent variables, which may not hold true for complex real-world behavioral data.
- **Outlier Sensitivity:** Both models are sensitive to outliers; a single anomaly (e.g., a student with high study hours but very low score due to illness) can skew the results.
- **Multicollinearity:** High correlation between independent variables (e.g., Assignment_Score and Midterm_Score) can distort the interpretability of feature importance.
- **Overfitting:** Without regularization, the models may memorize noise in the expanded dataset, leading to poor generalization on unseen data.

5. Methodology / Workflow

The experiment follows a standard Machine Learning pipeline:

1. **Data Preprocessing:**
   - Load the expanded_student_performance.csv dataset.
   - **Feature Engineering:** Create a binary Pass_Status column based on Final_Score for the Logistic Regression task.
   - Handle missing values (if any) and perform Feature Scaling (StandardScaler) to normalize input features.
2. **Data Splitting:** Divide the data into Training (80%) and Testing (20%) sets.
3. **Model Training:**
   - **Linear Regression:** Fit the model using Features vs. Final_Score.
   - **Logistic Regression:** Fit the model using Features vs. Pass_Status.
4. **Prediction:** Generate predictions on the test sets for both models.
5. **Evaluation:**
   - **Regression:** Calculate Mean Squared Error (MSE) and R-squared (R^2).

    - **Classification:** Compute Accuracy, Precision, Recall, and F1-Score.

6. Performance Analysis

Linear Regression Results:

- **Metric:** The R^2 score indicates how well the variance in Final_Score is explained by the study hours and interim grades. A score close to 1.0 implies a strong predictive capability.
- **Insight:** Students with high attendance and assignment scores consistently achieved higher final grades, validating the linear relationship.

Logistic Regression Results:

- **Metric:** Accuracy reflects the percentage of correctly identified Pass/Fail statuses.
- **Confusion Matrix:** Used to visualize True Positives (Correctly predicted Pass) vs. False Negatives (Incorrectly predicted Fail).
- **Insight:** The model successfully separates passing and failing students with high precision, demonstrating that study habits are a reliable predictor of academic success.

7. Hyperparameter Tuning

To optimize performance, the following hyperparameters can be tuned:

- **Learning Rate (α):** Controls the step size during gradient descent. Too large leads to divergence; too small leads to slow convergence.
- **Regularization (L1/L2):**
    - *Lasso (L1):* Encourages sparsity (feature selection), potentially removing less relevant features like 'Attendance' if it doesn't contribute significantly.
    - *Ridge (L2):* Penalizes large coefficients to prevent overfitting on the expanded dataset.
- **Max Iterations:** Ensures the solver converges within a reasonable time frame.

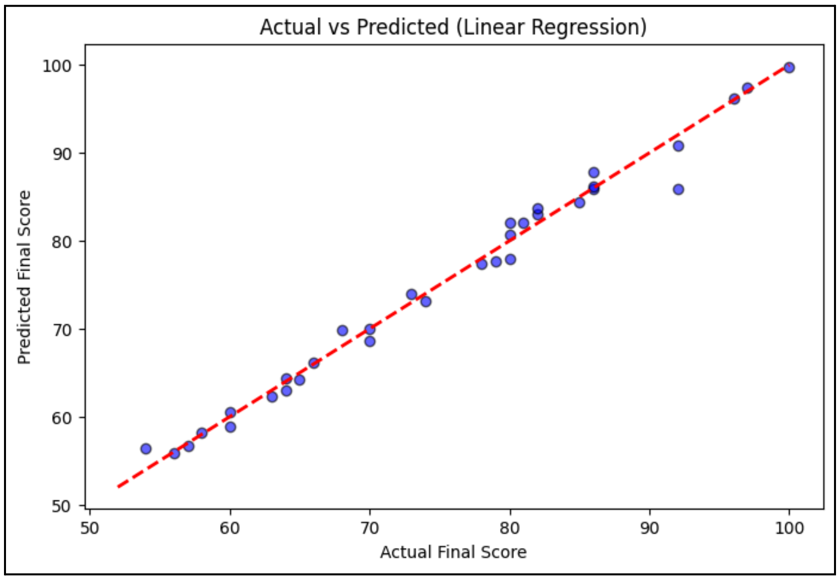
Exercise 1: Linear Regression

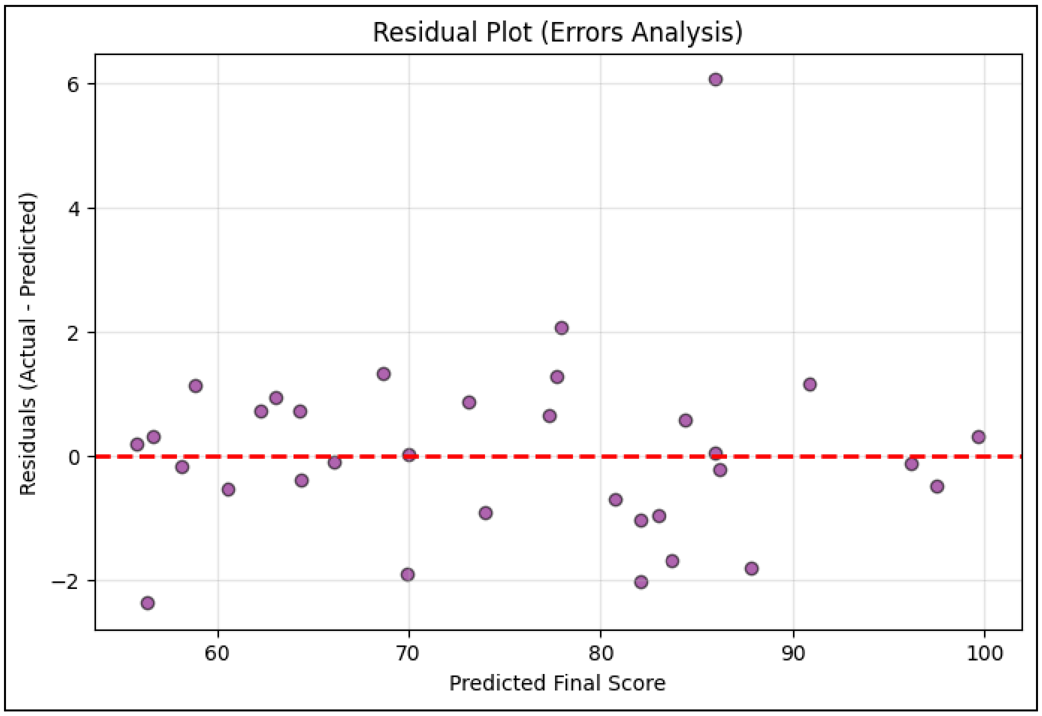
Code:

```
# Define Features (X) and Target (y)
X = df[['Hours_Studied', 'Attendance', 'Assignment_Score', 'Midterm_Score']]
y = df['Final_Score']
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.33, random_state=21)

# Initialize and Train the Model
lin_model = LinearRegression()
lin_model.fit(X_train, y_train)

# Make Predictions
y_pred = lin_model.predict(X_test)

# Evaluation
print("--- Linear Regression Results ---")
print(f"Mean Squared Error (MSE): {mean_squared_error(y_test, y_pred):.2f}")
print(f"R-squared (R2 Score): {r2_score(y_test, y_pred):.2f}")
print(f"Coefficients: {lin_model.coef_}")
print(f"Intercept: {lin_model.intercept_:.2f}")

plt.figure(figsize=(8, 5))
plt.scatter(y_test, y_pred, color='blue', edgecolors='k', alpha=0.6)
plt.plot([y.min(), y.max()], [y.min(), y.max()], 'r--', lw=2)
plt.xlabel('Actual Final Score')
plt.ylabel('Predicted Final Score')
plt.title('Actual vs Predicted (Linear Regression)')
plt.show()

# Calculate residuals
residuals = y_test - y_pred

plt.figure(figsize=(8, 5))
plt.scatter(y_pred, residuals, color='purple', edgecolors='k', alpha=0.6)
plt.axhline(y=0, color='r', linestyle='--', lw=2)  # Zero line for reference
plt.xlabel('Predicted Final Score')
plt.ylabel('Residuals (Actual - Predicted)')
plt.title('Residual Plot (Errors Analysis)')
plt.grid(True, alpha=0.3)
plt.show()
```

Output:

```
--- Linear Regression Results ---
Mean Squared Error (MSE): 2.26
R-squared (R2 Score): 0.99
Coefficients: [0.44531054 0.04058829 0.16006866 0.75653262]
Intercept: 2.26
```



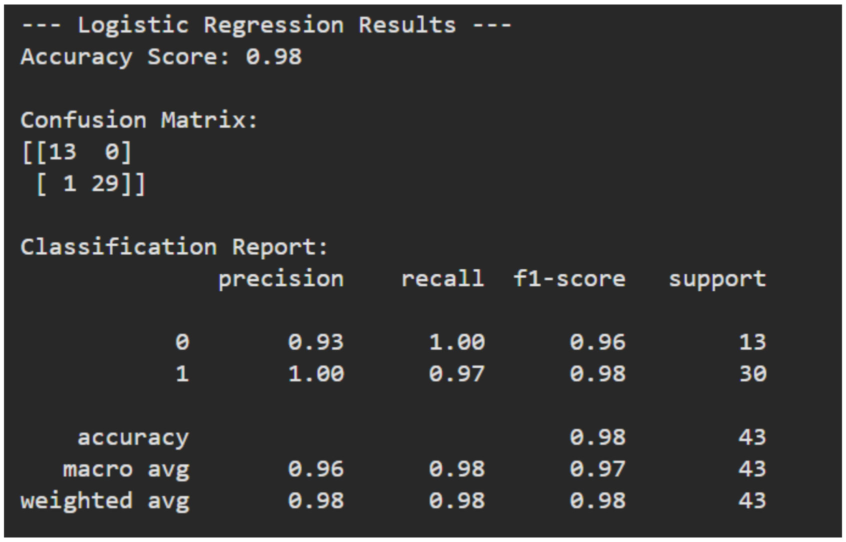


Exercise 2: Logistic Regression

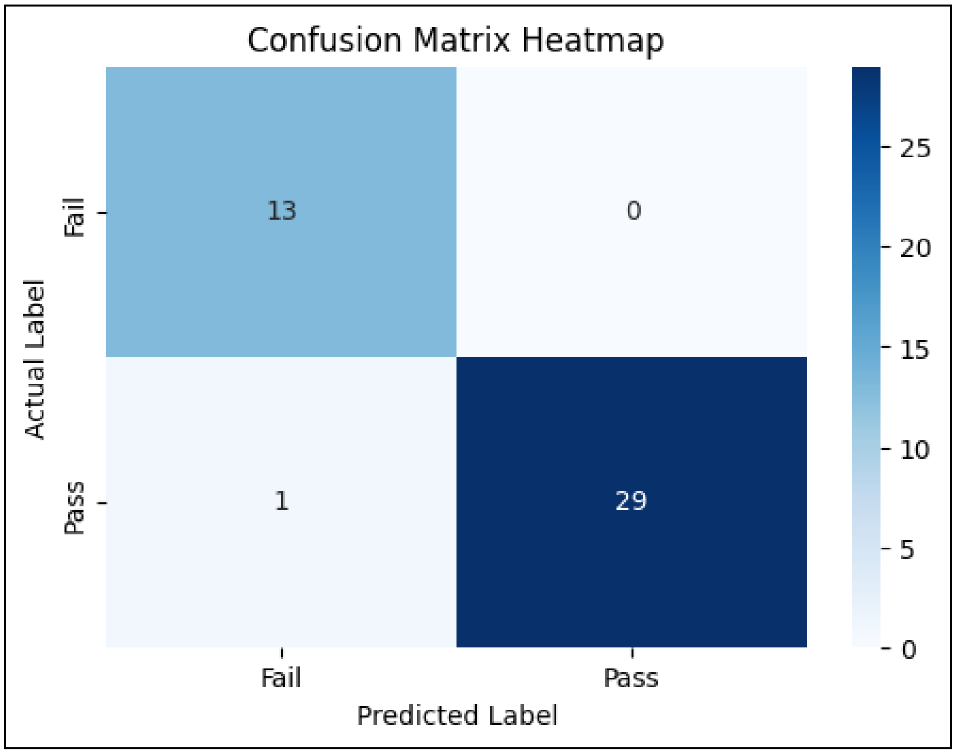
Code:

```
# Create a binary classification target
df['Pass'] = (df['Final_Score'] >= 70).astype(int)

# Define Features (X) and Target (y)
X_log = df[['Hours_Studied', 'Attendance', 'Assignment_Score', 'Midterm_Score']]
y_log = df['Pass']

# Split the data
X_train_log, X_test_log, y_train_log, y_test_log = train_test_split(X_log, y_log, test_size=0.43,
random_state=50)

# Initialize and Train the Model
log_model = LogisticRegression()
log_model.fit(X_train_log, y_train_log)

# Make Predictions
y_pred_log = log_model.predict(X_test_log)

# Evaluation
print("--- Logistic Regression Results ---")
print(f"Accuracy Score: {accuracy_score(y_test_log, y_pred_log):.2f}")
print("\nConfusion Matrix:")
print(confusion_matrix(y_test_log, y_pred_log))
print("\nClassification Report:")
print(classification_report(y_test_log, y_pred_log))

cm = confusion_matrix(y_test_log, y_pred_log)

plt.figure(figsize=(6, 4))
sns.heatmap(cm, annot=True, fmt='d', cmap='Blues',
            xticklabels=['Fail', 'Pass'],
            yticklabels=['Fail', 'Pass'])
plt.xlabel('Predicted Label')
plt.ylabel('Actual Label')
plt.title('Confusion Matrix Heatmap')
plt.show()
```

Output:



```
--- Logistic Regression Results ---
Accuracy Score: 0.98

Confusion Matrix:
[[13  0]
 [ 1 29]]

Classification Report:
              precision    recall  f1-score   support

           0       0.93      1.00      0.96        13
           1       1.00      0.97      0.98        30

    accuracy                           0.98        43
   macro avg       0.96      0.98      0.97        43
weighted avg       0.98      0.98      0.98        43
```



Conclusion

This experiment successfully implemented Linear and Logistic Regression on the expanded_student_performance.csv dataset. Linear Regression proved effective for estimating continuous student scores, while Logistic Regression provided robust binary classification for determining pass/fail outcomes. The use of evaluation metrics like MSE and Accuracy validated the reliability of these fundamental machine learning algorithms in analyzing academic performance data.