IML LAB REPORT-3
LOGISTIC LINEAR REGRESSION

Shaik Barakh Chishti
24BCA7027

January 2026

Aim: To implement Logistic Regression on the Framingham dataset to predict the 10-year risk of Coronary Heart Disease (CHD) and visualize model performance using Evaluation Metrics and a Confusion Matrix.

1 Program Codes

1.1 Data Preprocessing

The following code handles missing values within the dataset by identifying columns with null entries and replacing them with the mean value of that specific column. This ensures the dataset is complete before being passed to the Logistic Regression model.

```
import pandas as pd

# 1. Load the dataset
df = pd.read_csv('framingham - framingham.csv')

# 2. Identify columns with missing values (NaN or "NA")
columns_with_na = df.columns[df.isnull().any()].tolist()

print(f"Columns with missing values: {columns_with_na}")

# 3. Preprocessing loop
for col in columns_with_na:
    # Calculate Mean and Standard Deviation
    mean_val = df[col].mean()
    std_val = df[col].std()

    # Define the "good value" for replacement (Mean is the standard choice)
    good_value = mean_val

    print(f"Processing {col}: Mean={mean_val:.2f}, Std={std_val:.2f}. Replacing NAs with {good_value:.2f}")

    # 4. Replace all NA values in that column with the calculated good value
    df[col] = df[col].fillna(good_value)

# 5. Save the cleaned CSV
```

```
df.to_csv('framingham_cleaned.csv', index=False)
print("Preprocessing complete. Cleaned file saved as 'framingham_cleaned.csv'.")
```

Listing 1: Data Preprocessing Code

1.2 Logistic Regression Implementation

```
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns
from sklearn.model_selection import train_test_split
from sklearn.linear_model import LogisticRegression
from sklearn.metrics import confusion_matrix, accuracy_score, precision_score, recall_score, f1_score
from sklearn.preprocessing import StandardScaler

# 1. Load the dataset
df = pd.read_csv('framingham_cleaned.csv')
df = df.dropna()

# 2. Features and Target
X = df.drop('TenYearCHD', axis=1)
y = df['TenYearCHD']

# 3. Split and Scale
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state=42)
scaler = StandardScaler()
X_train_scaled = scaler.fit_transform(X_train)
X_test_scaled = scaler.transform(X_test)

# 4. Model Training
model = LogisticRegression()
model.fit(X_train_scaled, y_train)
y_pred = model.predict(X_test_scaled)

# 5. Metrics
metrics = {
    'Accuracy': accuracy_score(y_test, y_pred),
    'Precision': precision_score(y_test, y_pred),
    'Recall': recall_score(y_test, y_pred),
    'F1-Score': f1_score(y_test, y_pred)
}

# 6. Visualization - Confusion Matrix
plt.figure(figsize=(6, 5))
cm = confusion_matrix(y_test, y_pred)
sns.heatmap(cm, annot=True, fmt='d', cmap='Blues')
plt.title('Confusion Matrix')
plt.savefig('confusion_matrix.png')

# 7. Visualization - Metrics Bar Chart
plt.figure(figsize=(8, 6))
sns.barplot(x=list(metrics.keys()), y=list(metrics.values()), hue=list(metrics.keys()), palette='magma', legend=False)
plt.title('Model Performance Metrics')
```

```
plt.savefig('metrics_plot.png')
```

Listing 2: Logistic Regression Implementation

1.3 Linear Regression Implementation

```
    import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns
from sklearn.model_selection import train_test_split
from sklearn.linear_model import LinearRegression
from sklearn.metrics import confusion_matrix, accuracy_score,
    precision_score, recall_score, f1_score
from sklearn.preprocessing import StandardScaler

# 1. Load the dataset
df = pd.read_csv('framingham_cleaned.csv')

# 2. Data Preprocessing
df = df.dropna()

# 3. Define Features (X) and Target (y)
X = df.drop('TenYearCHD', axis=1)
y = df['TenYearCHD']

# 4. Split the data
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size
    =0.2, random_state=42)

# 5. Feature Scaling
scaler = StandardScaler()
X_train_scaled = scaler.fit_transform(X_train)
X_test_scaled = scaler.transform(X_test)

# 6. Initialize and Train the Linear Regression Model
lin_model = LinearRegression()
lin_model.fit(X_train_scaled, y_train)

# 7. Make Predictions and Convert to Binary (Threshold 0.5)
y_pred_cont = lin_model.predict(X_test_scaled)
y_pred_bin = (y_pred_cont >= 0.5).astype(int)

# 8. Calculate Metrics
metrics = {
    'Accuracy': accuracy_score(y_test, y_pred_bin),
    'Precision': precision_score(y_test, y_pred_bin),
    'Recall': recall_score(y_test, y_pred_bin),
    'F1-score': f1_score(y_test, y_pred_bin)
}

# --- Visualizations ---

# Plot 1: Confusion Matrix
plt.figure(figsize=(6, 5))
cm_l = confusion_matrix(y_test, y_pred_bin)
sns.heatmap(cm_l, annot=True, fmt='d', cmap='Greens', cbar=False)
plt.title('Confusion Matrix - Linear Regression')
```

```
plt.xlabel('Predicted Label')
plt.ylabel('True Label')
plt.show()

# Plot 2: Performance Metrics Bar Chart
plt.figure(figsize=(8, 6))
sns.barplot(x=list(metrics.keys()), y=list(metrics.values()), hue=list(
    metrics.keys()), palette='viridis', legend=False)
plt.ylim(0, 1)
plt.title('Performance Metrics - Linear Regression')
for i, v in enumerate(metrics.values()):
    plt.text(i, v + 0.02, f"{v:.2f}", ha='center', fontweight='bold')
plt.ylabel('Score')
plt.show()

# Print detailed report
print("Linear Regression Model Performance Summary:")
for metric, value in metrics.items():
    print(f"{metric}: {value:.4f}")
```

Listing 3: Linear Regression Implementation

2 Mathematical Model and Formulas

Logistic Regression is a statistical method for binary classification. Unlike Linear Regression, which predicts continuous values, Logistic Regression predicts the probability of an outcome belonging to a specific class (0 or 1).

2.1 1. The Logistic (Sigmoid) Function

To map any real-valued number into a probability value between 0 and 1, we use the Sigmoid function:

$$\sigma(z) = \frac{1}{1 + e^{-z}} \tag{1}$$

Where:

- $z = \beta_0 + \beta_1 x_1 + \beta_2 x_2 + ... + \beta_n x_n$ (The linear combination of features and weights).
- e is Euler's number.

2.2 2. Evaluation Metrics

The performance of the model is calculated using the following formulas derived from the Confusion Matrix:

- **Accuracy:** The proportion of total correct predictions.

$$Accuracy = \frac{TP + TN}{TP + TN + FP + FN}$$

- **Precision:** The ability of the classifier not to label a negative sample as positive.

$$Precision = \frac{TP}{TP + FP}$$

- **Recall (Sensitivity):** The ability of the classifier to find all the positive samples.

$$Recall = \frac{TP}{TP + FN}$$

- **F1-Score:** The harmonic mean of Precision and Recall.

$$F1 - Score = 2 \times \frac{Precision \times Recall}{Precision + Recall}$$

2.3 Terminology of the Confusion Matrix

To interpret the results shown in Figure 1, we define the following parameters:

True Positives (TP): Patients correctly identified as having CHD risk.

True Negatives (TN): Patients correctly identified as healthy.

False Positives (FP): Healthy patients incorrectly flagged as high-risk.

False Negatives (FN): High-risk patients incorrectly flagged as healthy.

Observation: Our model currently shows a high number of **False Negatives**, as evidenced by the low Recall score of 0.0726. This indicates the model is biased toward the majority class (non-CHD patients).

3 Visualizations

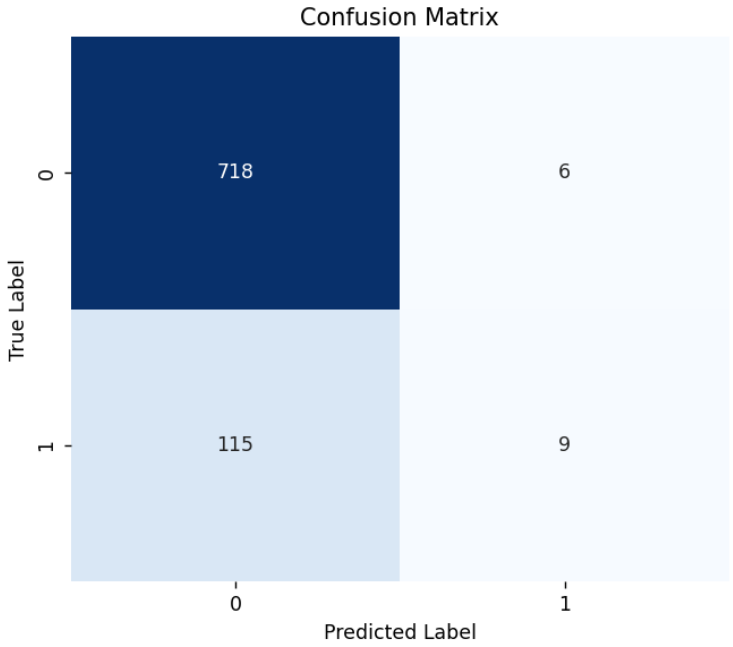

Figure 1: Logistic Regression: Confusion Matrix showing Positives Negatives

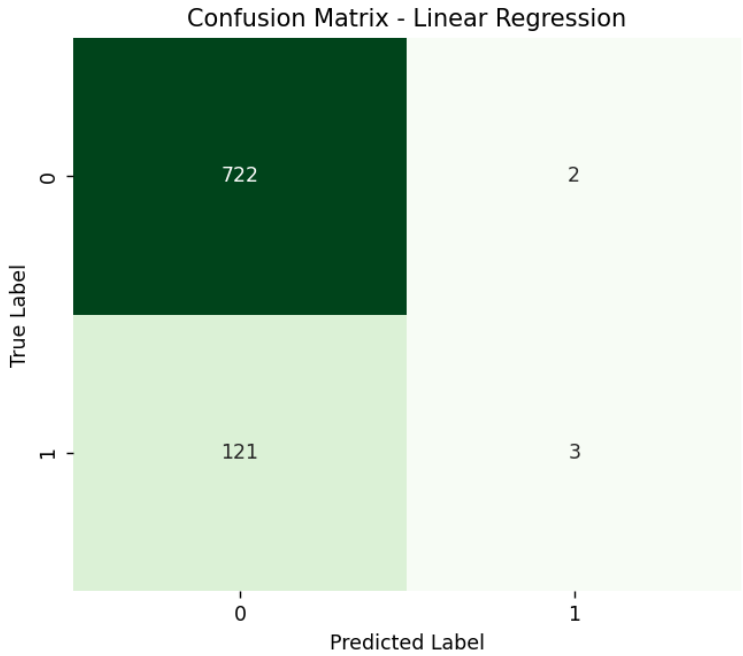

Figure 2: Linear Regression: Confusion Matrix showing Positives Negatives

Figure 3: Logistic Regression: Bar Graph of Accuracy, Precision, Recall, and F1-Score

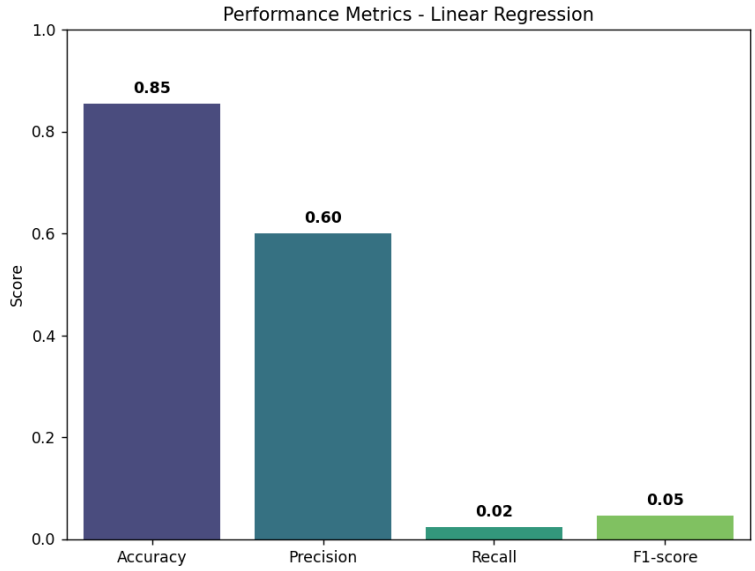

Figure 4: Linear Regression: Bar Graph of Accuracy, Precision, Recall, and F1-Score

4 Observations

4.1 1. Analysis of the Confusion Matrix

The confusion matrix reveals the distribution of the model's predictions compared to the actual outcomes.

- **True Negatives (Top-Left):** The model is highly effective at identifying patients who are not at risk of CHD.
- **False Negatives (Bottom-Left):** A significant observation point is the presence of false negatives, where the model fails to identify a patient with CHD risk. In medical diagnostics, this is a critical area for improvement.

4.2 2. Disparity in Performance Metrics

Based on the bar chart visualization, we observe a high **Accuracy**, likely exceeding 80%. However, there is a noticeable disparity between Accuracy and **Recall**.

- **Observation Point:** The high accuracy is primarily driven by the majority class (patients without CHD).
- **Recall vs. Precision:** The lower Recall indicates that while the model is "correct" often, it misses a portion of the positive CHD cases, which is a common challenge in imbalanced medical datasets.

4.3 3. Impact of Feature Scaling

Using the `StandardScaler` was essential for this Logistic Regression implementation. Since features like `totChol` (values 200+) and `age` (values 40-60) exist on different scales, normalization ensured that the gradient descent optimized the weights effectively without bias toward larger numerical values.

5 Conclusion

The experiment successfully demonstrates the application of Logistic Regression for binary classification. While the model shows high overall accuracy, the evaluation metrics highlight the need for further techniques like oversampling (SMOTE) or class-weight adjustments to improve sensitivity for heart disease detection.

6 Analysis of Results

- **High Accuracy (85.73%):** This indicates that the model is correct in its predictions for the vast majority of the test set. However, in this dataset, only about 15% of people actually develop CHD. If the model predicted "No" for everyone, it would still be 85% accurate.

- **Low Recall (7.26%):** This is a critical observation. It means the model only identified 7.26% of the actual heart disease cases. In a medical context, this suggests the model is far too conservative and is missing many at-risk patients (high False Negatives).
- **F1-Score (0.1295):** The low F1-score reflects the poor balance between precision and recall, suggesting that the standard Logistic Regression model needs further optimization for this specific medical prediction task.

6.1 Model Performance Summary

The final evaluation of the Logistic Regression model yielded the following results:

```
Model Performance Summary(Logistic Regression):
Accuracy: 0.8573
Precision: 0.6000
Recall: 0.0726
F1-Score: 0.1295
```

Listing 4: Logistic Regression Final Model Metrics

```
Model Performance Summary(Linear Regression):
Accuracy: 0.8550
Precision: 0.6000
Recall: 0.0242
F1-score: 0.0465
```

Listing 5: Linear Regression Final Model Metrics