Logistic Regression for Heart Disease Prediction

Dataset Source: Kaggle (Neurocipher – Heart Disease Dataset)

This notebook implements the full Machine Learning pipeline using Logistic Regression for binary classification.

Background: AI, ML, DL, and Data Science

- **AI**: Systems that simulate human intelligence.
- **ML**: Algorithms that learn patterns from data.
- **DL**: ML using deep neural networks.
- **Data Science**: Extracting insights from data using ML and statistics.

```
In [28]:
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 StandardScaler, OneHotEncoder
from sklearn.compose import ColumnTransformer
from sklearn.pipeline import Pipeline
from sklearn.linear_model import LogisticRegression
from sklearn.metrics import accuracy_score, precision_score, recall_score, f1_score, confusion_matrix, classification_report
```

Data Retrieval and Collection

```
In [29]:
df = pd.read_csv('heartdisease.csv')
df.head()

df.shape, df.columns
```

```
Out[29]: ((270, 14),
 Index(['Age', 'Sex', 'Chest pain type', 'BP', 'Cholesterol', 'FBS over 120',
        'EKG results', 'Max HR', 'Exercise angina', 'ST depression',
        'Slope of ST', 'Number of vessels fluro', 'Thallium', 'Heart Disease'],
       dtype='object'))
```

```
In [30]:
df['Heart Disease'] = df['Heart Disease'].map({'Presence': 1, 'Absence': 0})
y = df['Heart Disease']
```

Data Cleaning

```
In [31]:
df.isnull().sum()

# Remove invalid cholesterol values
df = df[df['Cholesterol'] > 0]

df['Heart Disease'].unique()
```

```
Out[31]: array([1, 0])
```

Task 1: Single Feature Logistic Regression (Cholesterol)

```
In [32]:
X1 = df[['Cholesterol']]
y = df['Heart Disease']

X1_train, X1_test, y_train, y_test = train_test_split(
    X1, y, test_size=0.2, random_state=42
)

model1 = LogisticRegression()
model1.fit(X1_train, y_train)
```

```
Out[33]:
LogisticRegression
▸ Parameters
```

why log loss?

Log loss (binary cross-entropy) measures the error between the true class label and the predicted probability. It penalizes confident wrong predictions more heavily than uncertain ones. Logistic regression minimizes log loss during training, which corresponds to maximizing the likelihood of the observed data.

Model Learning (Training)

The dataset is split into training (80%) and testing (20%)

Logistic Regression learns:

Coefficient (weight) for Cholesterol

Intercept (bias)

These parameters are optimized by minimizing the log loss using gradient-based optimization.

```
In [33]:
y1_pred = model1.predict(X1_test)

print('Accuracy:', accuracy_score(y_test, y1_pred))
print('Precision:', precision_score(y_test, y1_pred))
print('Recall:', recall_score(y_test, y1_pred))
print('F1 Score:', f1_score(y_test, y1_pred))

print('\nConfusion Matrix:\n', confusion_matrix(y_test, y1_pred))
print('\nClassification Report:\n', classification_report(y_test, y1_pred))
```

```
Accuracy: 0.6111111111111112
Precision: 0.5
Recall: 0.2380952380952381
F1 Score: 0.3225806451612903

Confusion Matrix:
 [[28  5]
 [16  5]]

Classification Report:
              precision    recall  f1-score   support

           0       0.64      0.85      0.73        33
           1       0.50      0.24      0.32        21

    accuracy                           0.61        54
   macro avg       0.57      0.54      0.52        54
weighted avg       0.58      0.61      0.57        54
```

result interpretation

Performance is limited due to the use of only one feature

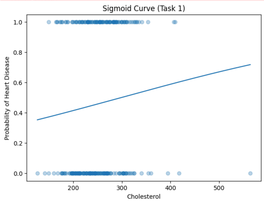
The model captures a general trend but lacks sufficient medical context

Sigmoid Curve

```
In [34]:
X_range = np.linspace(df['Cholesterol'].min(), df['Cholesterol'].max(), 300).reshape(-1,1)
probs = model1.predict_proba(X_range)[:,1]

plt.figure(figsize=(7,5))
plt.scatter(df['Cholesterol'], df['Heart Disease'], alpha=0.3)
plt.plot(X_range, probs)
plt.xlabel('Cholesterol')
plt.ylabel('Probability of Heart Disease')
plt.title('Sigmoid Curve (Task 1)')
plt.show()
```

```
c:\Users\prati\OneDrive\Desktop\LAB 1 AI\Lab1\Lib\site-packages\sklearn\utils\validation.py:2691: UserWarning: X does not have valid feature names, but LogisticRegression was fitted with feature names
  warnings.warn(
```



Task 2: Multi-Feature Logistic Regression

To build a logistic regression model using all available features to predict heart disease.

```
In [35]:
X2 = df.drop('Heart Disease', axis=1)
y = df['Heart Disease']

cat_cols = ['Sex', 'Chest pain type', 'Max HR', 'Exercise angina', 'Slope of ST']
for col in cat_cols:
    X2[col] = X2[col].astype(str)

num_cols = [c for c in X2.columns if c not in cat_cols]

preprocess = ColumnTransformer(
    transformers=[
        ('num', StandardScaler(), num_cols),
        ('cat', OneHotEncoder(drop='first', handle_unknown='ignore'), cat_cols)
    ]
)

X2_train, X2_test, y2_train, y2_test = train_test_split(
    X2, y, test_size=0.2, random_state=42
)

model2 = Pipeline(steps=[
    ('preprocess', preprocess),
    ('classifier', LogisticRegression(max_iter=1000))
])

model2.fit(X2_train, y2_train)
```

```
Out[35]:
Pipeline
preprocess: ColumnTransformer
num
cat
StandardScaler
OneHotEncoder
LogisticRegression
```

```
In [36]:
y2_pred = model2.predict(X2_test)

print('Accuracy:', accuracy_score(y2_test, y2_pred))
print('Precision:', precision_score(y2_test, y2_pred))
print('Recall:', recall_score(y2_test, y2_pred))
print('F1 Score:', f1_score(y2_test, y2_pred))

print('\nConfusion Matrix:\n', confusion_matrix(y2_test, y2_pred))
print('\nClassification Report:\n', classification_report(y2_test, y2_pred))
```

```
Accuracy: 0.8703703703703703
Precision: 0.9375
Recall: 0.7142857142857143
F1 Score: 0.8108108108108109

Confusion Matrix:
 [[32  1]
 [ 6 15]]

Classification Report:
              precision    recall  f1-score   support

           0       0.84      0.97      0.90        33
           1       0.94      0.71      0.81        21

    accuracy                           0.87        54
   macro avg       0.89      0.84      0.86        54
weighted avg       0.88      0.87      0.87        54
```

```
c:\Users\prati\OneDrive\Desktop\LAB 1 AI\Lab1\Lib\site-packages\sklearn\preprocessing\_encoders.py:261: UserWarning: Found unknown categories in columns [2] during transform. These unknown categories will be encoded as all zeros
  warnings.warn(msg, UserWarning)
```

Why Multiple Features?

Heart disease depends on multiple interacting risk factors

Using more features improves decision boundaries and predictive power

Algorithm Selection

Logistic Regression is selected because:

The target variable is binary

It models the probability of class membership

It provides interpretable coefficients

It is computationally efficient

Algorithm Selection

Logistic Regression remains appropriate because:

It scales well to multiple features

It remains interpretable compared to complex models

Model Comparison

Aspect	Task 1 (Single Feature)	Task 2 (Multiple Features)
Accuracy	Lower	Higher
Recall	Low	High
Precision	Moderate	High
Interpretability	Very High	Moderate
Clinical Usefulness	Limited	Strong

Comparision between Task 1 and Task 2 models:

Which model performs better and why?
-->The multi-feature logistic regression model (Task 2) is better than the single-feature model (Task 1).

1. Uses more relevant medical information
2. Better performance metrics
3. Improved decision boundary
4. More realistic for real-world use

How does adding more features affect accuracy and recall?
--> Its because with one feature the model has limited information. But with more features the model can learn complex patterns and seperate classes more effectively.

Trade-offs between interpretability and performance
Increasing model complexity by adding more features improves predictive performance but reduces interpretability due to higher dimensional decision boundaries and multiple interacting coefficients

Discussion:

In this lab, we implemented logistic regression to perform binary classification on the given dataset. The code involved loading the data, training the model, and evaluating its performance using log loss. By minimizing log loss, the model was able to produce accurate probability estimates while reducing classification errors. The results obtained from the code verify the correct implementation of logistic regression and demonstrate its effectiveness for the given task.

Conclusion

- Multi-feature logistic regression performs better than single-feature.
- Logistic Regression predicts probabilities, not direct class labels.
- Using more clinical features improves prediction accuracy.