In [ ]:
```python
import numpy as np
import pandas as pd
from sklearn.linear_model import LogisticRegression
```

In [ ]:
```python
data = pd.read_csv('framingham.csv')
print(data.shape)
data.head()
```

```
(4238, 16)
```

Out[ ]:

	male	age	education	currentSmoker	cigsPerDay	BPMeds	prevalentStroke	prevalentHyp	diabetes	totChol	sysBP	diaBP	BMI	heartRate	glucose	TenYearCHD
0	1	39	4.0	0	0.0	0.0	0	0	0	195.0	106.0	70.0	26.97	80.0	77.0	0
1	0	46	2.0	0	0.0	0.0	0	0	0	250.0	121.0	81.0	28.73	95.0	76.0	0
2	1	48	1.0	1	20.0	0.0	0	0	0	245.0	127.5	80.0	25.34	75.0	70.0	0
3	0	61	3.0	1	30.0	0.0	0	1	0	225.0	150.0	95.0	28.58	65.0	103.0	1
4	0	46	3.0	1	23.0	0.0	0	0	0	285.0	130.0	84.0	23.10	85.0	85.0	0

In [ ]:
```python
data = data.dropna(how="any", axis=0)
print(data.shape)
data.head()
```

```
(3656, 16)
```

Out[ ]:

	male	age	education	currentSmoker	cigsPerDay	BPMeds	prevalentStroke	prevalentHyp	diabetes	totChol	sysBP	diaBP	BMI	heartRate	glucose	TenYearCHD
0	1	39	4.0	0	0.0	0.0	0	0	0	195.0	106.0	70.0	26.97	80.0	77.0	0
1	0	46	2.0	0	0.0	0.0	0	0	0	250.0	121.0	81.0	28.73	95.0	76.0	0
2	1	48	1.0	1	20.0	0.0	0	0	0	245.0	127.5	80.0	25.34	75.0	70.0	0
3	0	61	3.0	1	30.0	0.0	0	1	0	225.0	150.0	95.0	28.58	65.0	103.0	1
4	0	46	3.0	1	23.0	0.0	0	0	0	285.0	130.0	84.0	23.10	85.0	85.0	0

In [ ]:
```python
# Chia dữ liệu thành features (X) và target (y)
X = data.drop('TenYearCHD', axis=1)
y = data['TenYearCHD']
```

In [ ]:
```python
# Thêm cột bias vào ma trận X
X = np.column_stack((np.ones(len(X)), X))

# Chia dữ liệu thành tập huấn luyện và tập kiểm tra
length = int(data.shape[0] * 0.7 // 1)
x_train, x_test = X[:length], X[length:]
y_train, y_test = y[:length], y[length:]
```

In [ ]:
```python
eta = .05
iterations = 10 ** 6
```

In [ ]:
```python
# Hàm sigmoid
def sigmoid(z):
    return 1 / (1 + np.exp(-z))

# Gradient descent
def logistic_regression(X, y, learning_rate, iterations = 10000, tol=1e-4):
    m, n = X.shape
    theta = np.zeros(n)
    for i in range(iterations):
        z = np.dot(X, theta)
        h = sigmoid(z)
        gradient = np.dot(X.T, (h - y)) / m
        if i % n == 0:
            if np.linalg.norm(theta - theta + learning_rate * gradient) < tol:
                return theta - learning_rate * gradient
        theta -= learning_rate * gradient

    return theta
```

In [ ]:
```python
# Thực hiện gradient descent để tìm các tham số theta
w = logistic_regression(x_train, y_train, eta, iterations)
```

```
C:\Users\Hoang Tu\AppData\Local\Temp\ipykernel_15232\2582513912.py:3: RuntimeWarning: overflow encountered in exp
  return 1 / (1 + np.exp(-z))
```

In [ ]:
```python
# Dự đoán trên tập kiểm tra
z = np.dot(x_test, w)
h = sigmoid(z)
y_pred = np.where(h >= 0.75, 1, 0)
```

```
C:\Users\Hoang Tu\AppData\Local\Temp\ipykernel_15232\2582513912.py:3: RuntimeWarning: overflow encountered in exp
  return 1 / (1 + np.exp(-z))
```

In [ ]:
```python
# Đánh giá mô hình
accuracy = np.mean(y_pred == y_test)
print("Độ chính xác:", accuracy)
```

```
Độ chính xác: 0.8486782133090246
```

In [ ]:
```python
model = LogisticRegression(max_iter=iterations)
model.fit(x_train, y_train)

sk_y_pred = model.predict(x_test)

# Đánh giá mô hình
accuracy = np.mean(sk_y_pred == y_test)
print("Độ chính xác:", accuracy)
```

```
Độ chính xác: 0.853236098450319
```

In [ ]:
```python
df = pd.DataFrame({'Real label': y_test, 'My solution': y_pred, 'Sklearn': sk_y_pred, })

print(df)
```

```
      Real label  My solution  Sklearn
2979           0            0        0
2980           0            0        0
2981           1            0        0
2982           0            0        0
2983           0            0        0
...          ...          ...      ...
4231           0            0        0
4232           1            0        0
4233           1            0        0
```

```
4234           0            0        0
4237           0            0        0

[1097 rows x 3 columns]
```