**1. Jelaskan cara kerja dari algoritma tersebut!**
**Jawab:**
Logistic Regression adalah model klasifikasi yang digunakan untuk memprediksi probabilitas suatu peristiwa atau kelas berdasarkan variabel input (fitur-fitur yang digunakan). Logistic Regression digunakan terutama untuk tugas klasifikasi biner. Cara kerjanya adalah sebagai berikut:

- Pertama-tama, Logistic Regression bekerja dengan cara yang mirip dengan regresi linear biasa. Kita menghitung kombinasi linear dari input features (fitur-fitur input), yaitu dengan menjumlahkan setiap fitur yang sudah dikalikan dengan bobot (parameter) tertentu.

  $$z = \beta_0 + \beta_1 X_1 + \beta_2 X_2 + \ldots + \beta_n X_n$$

  $X_1, X_2, ..., X_n$ adalah fitur-fitur dari data.

  $\beta_0, \beta_1, ..., \beta_n$ adalah bobot atau parameter yang harus kita pelajari dari data.

  $z$ adalah hasil dari kombinasi linear tersebut.

- Logistic Regression kemudian menggunakan fungsi sigmoid untuk mengubah hasil z tersebut menjadi sebuah nilai probabilitas antara 0 dan 1.

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

  $\sigma(z)$ adalah fungsi sigmoid.

  $p$ adalah probabilitas dari hasil prediksi

- Setelah mendapatkan probabilitas, kita bisa memutuskan kelas mana yang diprediksi oleh model dengan cara:
  - Jika probabilitas *p* lebih besar dari 0.5, kita memprediksi kelas 1.
  - Jika probabilitas *p* lebih kecil atau sama dengan 0.5, kita memprediksi kelas 0.
- Untuk mengukur seberapa baik model memprediksi data, kita menggunakan fungsi loss yang disebut Cross-Entropy Loss. Fungsi ini mengukur ketidaksesuaian antara prediksi model dan nilai sebenarnya.

  $$L(\theta) = -\frac{1}{m} \sum_{i=1}^{m} [y_i \log(\hat{y}_i) + (1 - y_i) \log(1 - \hat{y}_i)]$$

  $y_i$ adalah label asli (0 atau 1).

  $\hat{y}_i$ adalah probabilitas prediksi dari model.

  $m$ adalah jumlah data.

- Untuk menemukan bobot yang meminimalkan fungsi loss, kita menggunakan metode optimasi yang disebut Gradient Descent. Gradient Descent bekerja dengan memperbarui bobot *β* sedikit demi sedikit menuju arah yang mengurangi nilai loss.

  $$\beta_j := \beta_j - \alpha \cdot \frac{\partial L(\beta)}{\partial \beta_j}$$

$\alpha$ adalah learning rate, yaitu seberapa besar langkah pembaruan parameter.

$\frac{\partial L(\beta)}{\partial \beta_j}$ adalah gradien atau kemiringan fungsi loss terhadap bobot $\beta_j$.

- Proses optimasi ini diulang sampai model menemukan bobot-bobot yang optimal, yaitu ketika nilai loss tidak berubah lagi secara signifikan atau ketika mencapai batas jumlah iterasi tertentu.

**4. Bandingkan hasil evaluasi model pada nomor 2 dan 3, bagaimana hasil perbandingannya? Jika ada perbedaan, jelaskan alasannya!**
**Jawab:**

Menggunakan Implementasi From Scratch

```
import importlib
import logistic_regression
importlib.reload(logistic_regression)

from logistic_regression import LogisticRegression2

log = LogisticRegression2(learning_rate=0.13, num_iterations=1000, regularization='l2', lambda_=0.01, loss_function="cross_entropy")

# Using holdout validation
log.fit(X_train, y_train)
y_pred = log.predict(X_test)
ho_accuracy = accuracy_score(y_test, y_pred)
print(f"Accuracy: {ho_accuracy}")

# Using k-fold cross-validation
cv_accuracy = cross_val_score(log, X, y, cv=kfold, scoring="accuracy").mean()
print(f"Cross-validation Accuracy: {cv_accuracy}")
```

✓ 0.5s

```
Accuracy: 0.7868852459016393
Cross-validation Accuracy: 0.8376502732240437
```

Menggunakan Implementasi dari Library

```
log_lib = LogisticRegression(penalty='l2', solver='lbfgs', max_iter=1000, C=1/0.01)

# Using holdout validation
log_lib.fit(X_train, y_train)
y_pred_lib = log_lib.predict(X_test)
ho_accuracy_lib = accuracy_score(y_test, y_pred_lib)
print(f'Accuracy: {ho_accuracy_lib}')

# Using k-fold cross-validation
cv_accuracy_lib = cross_val_score(log_lib, X, y, cv=kfold, scoring="accuracy").mean()
print(f'Cross-validation Accuracy: {cv_accuracy_lib}')
```

✓ 0.0s

```
Accuracy: 0.7868852459016393
Cross-validation Accuracy: 0.8343715846994536
```

Kedua model memberikan hasil akurasi yang hampir sama, baik pada holdout validation maupun pada cross-validation. Adapun perbedaan dapat disebabkan karena Scikit-Learn menggunakan optimasi yang lebih canggih (L-BFGS) yang cenderung lebih cepat mencapai konvergensi optimal dibandingkan dengan gradient descent biasa dan perbedaan kecil dalam perhitungan numerik antara kedua implementasi.

**5. Jelaskan improvement apa saja yang bisa Anda lakukan untuk mencapai hasil yang lebih baik dibandingkan dengan hasil yang Anda punya saat ini! Improvement yang**

**dimaksud tidak terbatas pada bagaimana algoritma diimplementasikan, namun juga mencakup tahap sebelum modeling and validation.**

**Jawab:**

- Menggunakan teknik optimasi yang lebih canggih.
- Mencoba terapkan teknik Feature Selection atau Dimensionality Reduction untuk mengurangi noise dalam data yang bisa meningkatkan performa model.