Francesco Michael Kusuma

13522038

Logistic Regression

Cara Kerja:

1. Inisiasi *learning rate*, jumlah iterasi, *regularization term, dan loss function* yang akan digunakan
2. Hitung output model menggunakan fungsi sigmoid

   $\sigma(z) = \frac{1}{1+ e^{-z}}$, dengan $z = X \cdot w + b$
3. Tentukan nilai *loss* menggunakan *loss function* yang telah ditentukan.
4. Update parameter w dan b menggunakan *Gradient Descent atau Newthon's method.*
5. Ulang langkah 2-4 hingga konvergen atau jumlah iterasi terpenuhi.

Perbandingan:

- *Scratch*
  - *Gradient descent*

```
[25]: from logisticRegression import LogisticRegressionScratch

logreg_scratch_gd = LogisticRegressionScratch(learning_rate = 0.01, iterations = 1000, regularization=None, loss_function='log_loss')
logreg_scratch_gd.fit(X_train, y_train)
y_pred_logreg_scratch_gd = logreg_scratch_gd.predict(X_test)

validate_model(logreg_scratch_gd, method_name="Logistic Regression with Gradient Descent from Scratch")
```

```
Hold-Out Validation (Logistic Regression with Gradient Descent from Scratch):
F1 Score: 0.868421052631579
              precision    recall  f1-score   support

           0       0.89      0.99      0.93        72
           1       0.97      0.79      0.87        42

    accuracy                           0.91       114
   macro avg       0.93      0.89      0.90       114
weighted avg       0.92      0.91      0.91       114


K-Fold Cross-Validation (Logistic Regression with Gradient Descent from Scratch):
F1 Scores for each fold: [0.90625, 0.88, 0.9295774647887324, 0.5901639344262295, 0.7936507936507936]
Mean F1 Score: 0.8199284385731511
Standard Deviation of F1 Score: 0.12374583257467736
```

- ## *Hinge Loss*

[26]:
```
logreg_scratch_hinge = LogisticRegressionScratch(learning_rate = 0.01, iterations = 1000, regularization=None, loss_function='hinge_loss')
logreg_scratch_hinge.fit(X_train, y_train)
y_pred_logreg_scratch_hinge = logreg_scratch_hinge.predict(X_test)

validate_model(logreg_scratch_hinge, method_name="Logistic Regression with Hinge Loss from Scratch")
```

```
Hold-Out Validation (Logistic Regression with Hinge Loss from Scratch):
F1 Score: 0.868421052631579
              precision    recall  f1-score   support

           0       0.89      0.99      0.93        72
           1       0.97      0.79      0.87        42

    accuracy                           0.91       114
   macro avg       0.93      0.89      0.90       114
weighted avg       0.92      0.91      0.91       114


K-Fold Cross-Validation (Logistic Regression with Hinge Loss from Scratch):
F1 Scores for each fold: [0.90625, 0.88, 0.9295774647887324, 0.5901639344262295, 0.7936507936507936]
Mean F1 Score: 0.8199284385731511
Standard Deviation of F1 Score: 0.12374583257467736
```

- ## *L1 Regularization*

[27]:
```
logreg_scratch_l1 = LogisticRegressionScratch(learning_rate = 0.01, iterations = 1000, regularization='l1', loss_function='hinge_loss')
logreg_scratch_l1.fit(X_train, y_train)
y_pred_logreg_scratch_l1 = logreg_scratch_l1.predict(X_test)

validate_model(logreg_scratch_l1, method_name="Logistic Regression with L1 Regularization from Scratch")
```

```
Hold-Out Validation (Logistic Regression with L1 Regularization from Scratch):
F1 Score: 0.8974358974358975
              precision    recall  f1-score   support

           0       0.91      0.99      0.95        72
           1       0.97      0.83      0.90        42

    accuracy                           0.93       114
   macro avg       0.94      0.91      0.92       114
weighted avg       0.93      0.93      0.93       114


K-Fold Cross-Validation (Logistic Regression with L1 Regularization from Scratch):
F1 Scores for each fold: [0.90625, 0.9117647058823529, 0.9295774647887324, 0.2857142857142857, 0.8064516129032258]
Mean F1 Score: 0.7679516138577194
Standard Deviation of F1 Score: 0.2449353609905441
```

- ## *L2 Regularization*

[28]:
```
logreg_scratch_l2 = LogisticRegressionScratch(learning_rate = 0.01, iterations = 1000, regularization='l2', loss_function='hinge_loss')
logreg_scratch_l2.fit(X_train, y_train)
y_pred_logreg_scratch_l2 = logreg_scratch_l2.predict(X_test)

validate_model(logreg_scratch_l2, method_name="Logistic Regression with L2 Regularization from Scratch")
```

```
Hold-Out Validation (Logistic Regression with L2 Regularization from Scratch):
F1 Score: 0.868421052631579
              precision    recall  f1-score   support

           0       0.89      0.99      0.93        72
           1       0.97      0.79      0.87        42

    accuracy                           0.91       114
   macro avg       0.93      0.89      0.90       114
weighted avg       0.92      0.91      0.91       114


K-Fold Cross-Validation (Logistic Regression with L2 Regularization from Scratch):
F1 Scores for each fold: [0.90625, 0.88, 0.9295774647887324, 0.6, 0.7936507936507936]
Mean F1 Score: 0.8218956516879052
Standard Deviation of F1 Score: 0.12010212028781977
```

- *Newton's Method*

Newton's Method

```
[29]: logreg_scratch_n = LogisticRegressionScratch(learning_rate = 0.01, iterations = 1000, method='newton')
      logreg_scratch_n.fit(X_train, y_train)
      y_pred_logreg_scratch_n = logreg_scratch_n.predict(X_test)

      validate_model(logreg_scratch_n, method_name="Logistic Regression with Newton's Method from Scratch")
```

```
Hold-Out Validation (Logistic Regression with Newton's Method from Scratch):
F1 Score: 0.8505747126436781
              precision    recall  f1-score   support

           0       0.93      0.89      0.91        72
           1       0.82      0.88      0.85        42

    accuracy                           0.89       114
   macro avg       0.87      0.88      0.88       114
weighted avg       0.89      0.89      0.89       114


K-Fold Cross-Validation (Logistic Regression with Newton's Method from Scratch):
F1 Scores for each fold: [0.9428571428571428, 0.9552238805970149, 0.9428571428571428, 0.9014084507042254, 0.8888888888888888]
Mean F1 Score: 0.9262471011808829
Standard Deviation of F1 Score: 0.026092279691359323
```

- *Library*

```
[30]: from sklearn.linear_model import LogisticRegression

      logreg = LogisticRegression()
      logreg.fit(X_train, y_train)
      y_pred_logreg = logreg.predict(X_test)

      validate_model(logreg, method_name="Logistic Regression from Library")
```

```
Hold-Out Validation (Logistic Regression from Library):
F1 Score: 0.8860759493670886
              precision    recall  f1-score   support

           0       0.91      0.97      0.94        72
           1       0.95      0.83      0.89        42

    accuracy                           0.92       114
   macro avg       0.93      0.90      0.91       114
weighted avg       0.92      0.92      0.92       114


K-Fold Cross-Validation (Logistic Regression from Library):
F1 Scores for each fold: [0.9393939393939394, 0.9846153846153847, 0.9142857142857143, 0.9014084507042254, 0.8387096774193549]
Mean F1 Score: 0.9156826332837238
Standard Deviation of F1 Score: 0.0478424285527039
```

Dari implementasi secara *scratch dan library*, terlihat bahwa penggunaan *library* memiliki nilai *F1 score* yang lebih tinggi. Akan tetapi, implementasi secara *scratch* pada *L1 Regularization, F1 Score* yang dihasilkan lebih tinggi disbanding dengan implementasi menggunakan *library*.

Improvement:

Improvement yang dapat dilakukan pada algoritma *Logistic Regression* secara *scratch* dapat dilakukan dengan memperbaiki *learning rate*, jumlah iterasi, *threshold* yang digunakan, dan sebagainya.