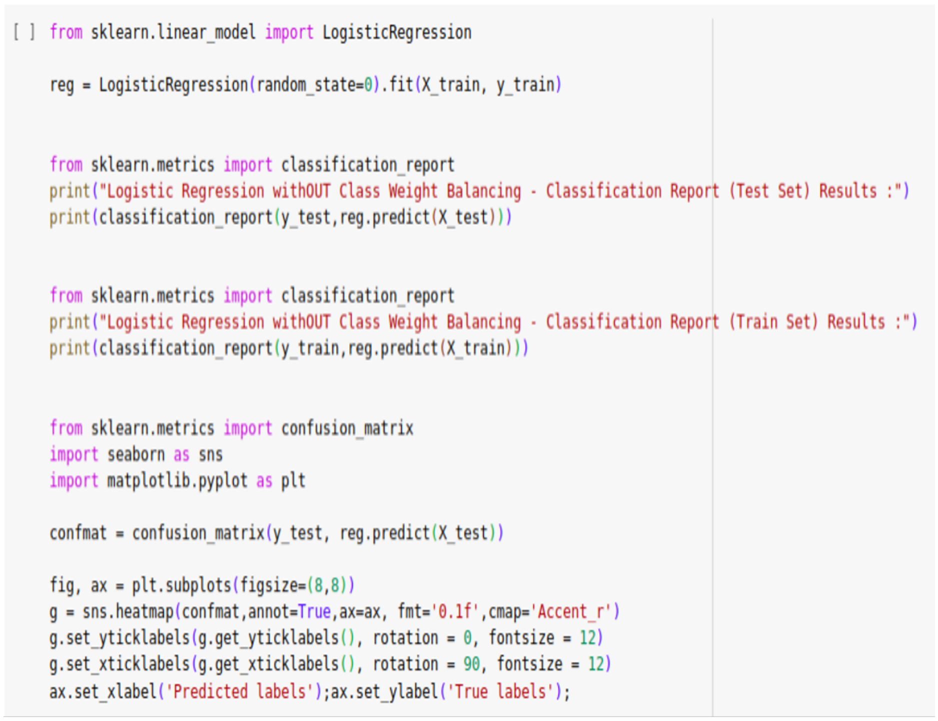
A2.51 Logistic Regression Without Class Weight Balancing

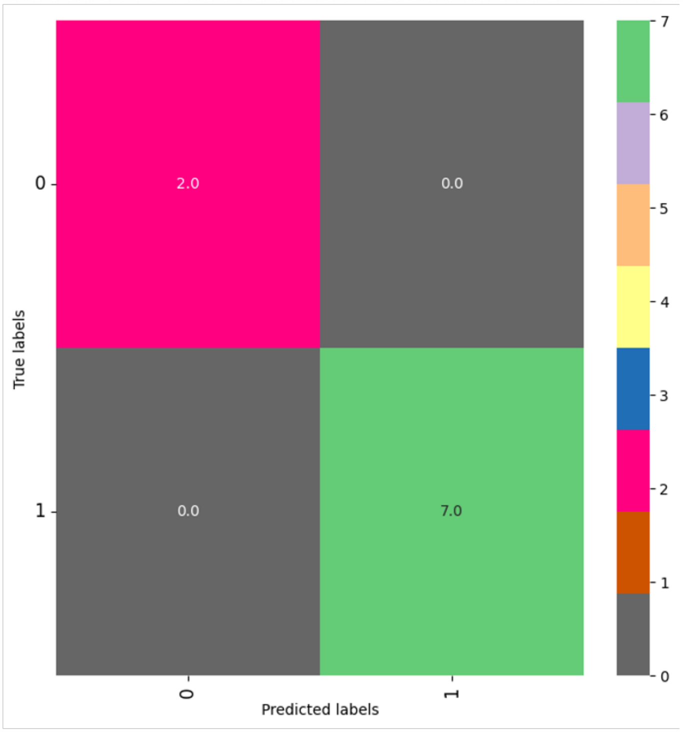


```
[ ] from sklearn.linear_model import LogisticRegression

    reg = LogisticRegression(random_state=0).fit(X_train, y_train)


    from sklearn.metrics import classification_report
    print("Logistic Regression withOUT Class Weight Balancing - Classification Report (Test Set) Results :")
    print(classification_report(y_test,reg.predict(X_test)))


    from sklearn.metrics import classification_report
    print("Logistic Regression withOUT Class Weight Balancing - Classification Report (Train Set) Results :")
    print(classification_report(y_train,reg.predict(X_train)))


    from sklearn.metrics import confusion_matrix
    import seaborn as sns
    import matplotlib.pyplot as plt

    confmat = confusion_matrix(y_test, reg.predict(X_test))

    fig, ax = plt.subplots(figsize=(8,8))
    g = sns.heatmap(confmat,annot=True,ax=ax, fmt='0.1f',cmap='Accent_r')
    g.set_yticklabels(g.get_yticklabels(), rotation = 0, fontsize = 12)
    g.set_xticklabels(g.get_xticklabels(), rotation = 90, fontsize = 12)
    ax.set_xlabel('Predicted labels');ax.set_ylabel('True labels');
```

This code is about creating a logistic regression model, training it, and then evaluating its performance using various metrics. The code is intended to give a detailed evaluation of the logistic regression model's performance, first on the test data and then on the training data, followed by a visual representation of the confusion matrix for the test data:

1. `from sklearn.linear_model import LogisticRegression`: This line imports the `LogisticRegression` class from the `sklearn` library. Logistic regression is a linear model for classification.
2. `reg = LogisticRegression(random_state=0).fit(X_train, y_train)`: This line initializes a logistic regression model with a specified random state (for reproducibility) and fits it to the training data (`X_train` and `y_train`).
4-6. `from sklearn.metrics import classification_report`: The classification report function is imported, which will be used later to display precision, recall, and F1-score for each class.
7. `print(classification_report(y_test,reg.predict(X_test)))`: The performance of the logistic regression model is evaluated on the test set, and the results (precision, recall, F1-score, etc.) are printed.
9-11. `print(classification_report(y_train,reg.predict(X_train)))`: The performance of the logistic regression model is evaluated on the training set. This step is useful for comparing training vs. test performance, which can hint at issues like overfitting if there's a big discrepancy between the two.
13. `from sklearn.metrics import confusion_matrix`: This imports the function to calculate the confusion matrix. A confusion matrix is used to understand the true positives, true negatives, false positives, and false negatives.
14-16. These lines are about setting up the necessary tools for plotting. `seaborn` is imported as `sns` for advanced plotting and `matplotlib.pyplot` is imported as `plt` for basic plotting utilities.
18. `confmat = confusion_matrix(y_test, reg.predict(X_test))`: This line computes the confusion matrix for the test data based on the model's predictions.
20. `fig, ax = plt.subplots(figsize=(8,8))`: Initializes a new figure and axis object for plotting, with specified size.
21. The `sns.heatmap()` function is used to plot the confusion matrix as a heatmap. Parameters:
   - `annot=True`: Annotate each cell with its value.
   - `fmt='0.1f'`: Format string for annotations (one decimal place).
   - `cmap='Accent_r'`: Use the 'Accent' colormap in reverse order.
22-23. These lines rotate the y-tick labels to horizontal and the x-tick labels to vertical and set the font size for better visibility and clarity.
24. Labels for x and y axes are set. The x-axis represents the predicted labels by the model, and the y-axis represents the actual labels.

```
Logistic Regression withOUT Class Weight Balancing - Classification Report (Test Set) Results :
              precision    recall  f1-score   support

           0       1.00      1.00      1.00         2
           1       1.00      1.00      1.00         7

    accuracy                           1.00         9
   macro avg       1.00      1.00      1.00         9
weighted avg       1.00      1.00      1.00         9

Logistic Regression withOUT Class Weight Balancing - Classification Report (Train Set) Results :
              precision    recall  f1-score   support

           0       1.00      1.00      1.00         4
           1       1.00      1.00      1.00        16

    accuracy                           1.00        20
   macro avg       1.00      1.00      1.00        20
weighted avg       1.00      1.00      1.00        20
```



The results presented seem to be the classification reports of a logistic regression model for two datasets (possibly a test set and a training set). Let's break down the results:

First Dataset:

- **Class 0**:
  - **Precision:** 1.00 - When the model predicts an instance as class 0, it's correct 100% of the time.
  - **Recall:** 1.00 - Out of all the actual class 0 instances, the model correctly identifies all of them.
  - **F1-score:** 1.00 - The harmonic mean of precision and recall is perfect for this class.
  - **Support:** 2 - There are 2 instances of class 0 in this dataset.

- **Class 1**:
  - **Precision:** 1.00 - Every prediction the model makes as class 1 is correct.
  - **Recall:** 1.00 - The model correctly identifies all instances of class 1.
  - **F1-score:** 1.00 - The harmonic mean of precision and recall for class 1 is also perfect.
  - **Support:** 7 - There are 7 instances of class 1 in this dataset.

- **Overall Metrics**:
  - **Accuracy:** 1.00 - All predictions made by the model for this dataset are correct.
  - **Macro avg:** Averages the scores for each class:
    - Precision, Recall, and F1-score are all 1.00.
  - **Weighted avg:** Considers the class distribution for the averaging:
    - Precision, Recall, and F1-score are all 1.00.

Second Dataset:

- **Class 0**:
  - **Precision:** 1.00 - When the model predicts an instance as class 0, it's always correct.
  - **Recall:** 1.00 - All actual class 0 instances are correctly identified by the model.
  - **F1-score:** 1.00 - Perfect balance between precision and recall for class 0.
  - **Support:** 4 - There are 4 instances of class 0 in this dataset.

- **Class 1**:
  - **Precision:** 1.00 - Every time the model predicts class 1, it's accurate.
  - **Recall:** 1.00 - The model captures all class 1 instances correctly.
  - **F1-score:** 1.00 - Perfect balance between precision and recall for this class.
  - **Support:** 16 - There are 16 instances of class 1 in this dataset.

- **Overall Metrics**:
  - **Accuracy:** 1.00 - Every single prediction made by the model for this dataset