Corrección Sobre
Momento de Retroalimentación: Módulo 2 Análisis y Reporte sobre el desempeño del modelo. (Portafolio Análisis)

Este análisis sobre el el desempeño del modelo sobre un dataset en específico, se basa en la implementación de Logistic Regression sobre un dataset de Diabetes. El modelo es capaz de predecir si una persona tiene o no tiene diabetes en base a ciertas características médicas.

La implementación de este algoritmo puede ser encontrado en: https://github.com/Caceres-A01706972/TC3006C-Mod2-Framework/blob/main/logistic_framework.py

Separación y evaluación del modelo con un conjunto de prueba y un conjunto de validación (Train/Test/Validation):

En la implementación del modelo, se realiza una división de los datos de iris.data en conjuntos de entrenamiento y prueba utilizando la función de **train_test_split** de *Scikit-Learn*.

```
# Split the dataset into training and testing sets
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state=42)
```

La función de *Scikit-Learn* toma los features, el target (en este caso la etiqueta de Outcome), el test_size que especifica la proporción del conjunto de datos que se va a utilizar como conjunto de test. En este caso, se estableció un test_size de 0.2 (significa que el 20% del conjunto de datos será utilizado para ser el test, mientras que el otro 80% será el test).

Diagnóstico y explicación del grado de bias o sesgo (bajo, medio, alto):

El modelo tuvo los siguientes resultados:

```
Training Accuracy: 77.04%
Testing Accuracy: 75.32%
```

Esto nos indica que el modelo si tiene cierto sesgo, ya que el rendimiento en el conjunto de training es ligeramente mejor al conjunto de training. Con esto podemos decir que el modelo tiene un sesgo medio, el cual significa que si existe cierta capacidad para generalizar, pero aún hay un margen de mejora para que si se pueda llegar a generalizar. Lo ideal sería tener un sesgo bajo, lo que significa que ambos conjuntos tengan un rendimiento igual.

Diagnóstico y explicación del grado de varianza (bajo, medio, alto):

La diferencia que existe entre el rendimiento del conjunto de training y el conjunto de testing es del 1.72%:

```
Training Accuracy: 77.04%
Testing Accuracy: 75.32%
```

Como no es significativamente grande, la varianza es moderadamente baja. En el caso de que tuviéramos una diferencia significativa entre el conjunto de training con el conjunto de test, tuviéramos una varianza alta lo que nos indicaría que hay overfitting. Pero en este caso, la varianza es moderadamente baja entonces lo que nos indica que existe una buena capacidad de generalización del modelo

Diagnóstico y explicación del nivel de ajuste del modelo (underfit, fit, overfit):

Los resultados del classification report son los siguientes:

```
Classification Report:
              precision    recall  f1-score   support

           0       0.80      0.83      0.81        99
           1       0.67      0.62      0.64        55

    accuracy                           0.75       154
   macro avg       0.73      0.72      0.73       154
weighted avg       0.75      0.75      0.75       154
```

El modelo no muestra signos evidentes de underfitting o de overfitting. Los valores de la precisión, el recall y el f1-score en el training son razonablemente buenos y no indican algún overfitting o underfitting extremo. No es perfecto, pero se puede decir que el modelo está relativamente bien equilibrado en términos de capacidad de ajuste y que tiene sus areas de mejora.

Uso de técnicas de regularización o ajuste de parámetros para mejorar el desempeño del modelo:

Como el modelo no muestra un sesgo o una varianza extremadamente altos, podríamos decir que esta en buen camino. Sin embargo, como se mencionó anteriormente, el modelo no es perfecto y se puede llegar a mejorar su rendimiento.

Para mejorar el modelo se pueden llegar a ocupar diferentes técnicas. Una de ellas es la regularización. En este caso se aplicará la regularización de L2 (también conocida como Ridge regularization). Para aplicar la regularización L2 en scikit-learn, simplemente necesitamos configurar el parámetro C en el modelo LogisticRegression.

```python
# Aplicar regularización L2 con el parámetro C
model = LogisticRegression(max_iter=max_iter, C=0.1)
model.fit(X_train, y_train)
```

Sin embargo, al hacer esto, nuestro resultado no cambió. Pero modificando el valor de C a un valor más pequeño podemos incrementar la fuerza de la regularización.

```python
# Aplicar regularización L2 con el parámetro C
model = LogisticRegression(max_iter=max_iter, C=0.01)
model.fit(X_train, y_train)
return model
```

Y ahora sí, podemos ver una mejora en la precisión de nuestro modelo:

```
Training Accuracy: 77.52%
Testing Accuracy: 76.62%
```

Si ya la precisión no llega a cambiar, puede ser porque el modelo ya está generalizado bien y no muestra signos evidentes de overfitting en los datos de prueba, entonces la regularización podría ya no ser necesaria y no tener un impacto en el rendimiento.

Otra manera en la que podríamos mejorar el rendimiento del modelo es mejorando la selección de las características. Podríamos investigar si algunas características no contribuyen significativamente al modelo y considerar eliminarlas. Esto simplificará el modelo y posiblemente mejoraría su rendimiento.