Assignment Code: DA-AG-011

Logistic Regression | **Assignment**

Question 1: What is Logistic Regression, and how does it differ from Linear

Regression?

Answer: Logistic Regression is a statistical method used for binary and multiclass classification problem. It models the probability that a given input belongs to a particular class. The output is always between 0 and 1 , which makes it suitable for predicting probabilities.

Difference between Logistic and Linear Regression:

Aspect	**Logistic Regression**	**Linear Regression**
Purpose	Classification	Regression
Output	Probability (0 to 1)	Continuous value
Function Used	Sigmoid/Logistic function	Linear function
Linearity	Models a linear decision boundary	Models a linear relationship between variables
Target Variable	Categorical	Continuous

Question 2: Explain the role of the Sigmoid function in logistic Regression.

Answer: The Sigmoid function is used in Logistic Regression to map any real – valued number to a probability between 0 and 1. It helps to interpret the output of the linear equation as a probability.

The sigmoid function is defined as:

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

Where z is the linear combination of input features and weights. The output of this function is used to classify the input data into different categories.

Question 3: What is Regularization in Logistic Regression and why is it needed?

Answer: Regularization is a technique used to prevent overfitting by penalizing large coefficients in the model.

In Logistic Regression, regularization adds a penalty term to the cost function:

- L1 (Lasso) Regularization adds the absolute value of coefficients.

- L2 (Ridge) Regularization adds the square of the coefficients.

This encourages the model to find simpler solutions that generalize better to unseen data.

Regularization is especially useful when dealing with high-dimensional data or multicollinearity.

Question 4: What are some common evaluation metrics for classification models, and why are they important?

Answer: Common evaluation metrics for classification:

- Accuracy: Proportion of correct predictions.
- Precision: Correct positive prediction / Total actual positives.
- Recall (Sensitivity): Correct positive prediction/ Total actual positives.
- F1 Score: Harmonic mean of precision and recall.
- Confusion Matrix: Summary of prediction results on classification.
- ROC-AUC Score: Measures the ability of a classifier to distinguish classes.

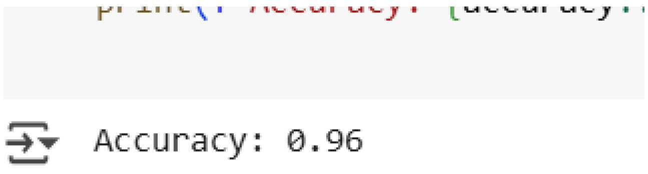
These metrics are important because they highlight different aspects of model performance, especially in imbalanced datasets where accuracy alone can be misleading.

Question 5: Python Program – Train Logistic Regression and Print Accuracy

Answer:

```
from sklearn.datasets import load_breast_cancer
from sklearn.model_selection import train_test_split
from sklearn.linear_model import LogisticRegression
from sklearn.metrics import accuracy_score

# Load dataset
data = load_breast_cancer()
X = data.data
y = data.target

# Train-test split
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state=42)

# Train logistic regression
model = LogisticRegression(max_iter=10000)
model.fit(X_train, y_train)

# Predict and evaluate
y_pred = model.predict(X_test)
```

```
accuracy = accuracy_score(y_test, y_pred)
print(f"Accuracy: {accuracy:.2f}")
```

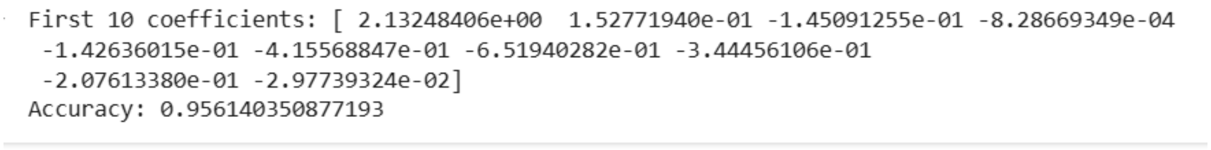


```
Accuracy: 0.96
```

Question 6: Write a Python program to train a logistic Regression model using L2 regularization (Ridge) and print the model coefficients and accuracy.

Answer:

```
# Train logistic regression with L2 regularization

from sklearn.linear_model import LogisticRegression

from sklearn.metrics import accuracy_score


model_l2 = LogisticRegression(penalty='l2', C=1.0, solver='liblinear', max_iter=1000)

model_l2.fit(X_train, y_train)

print('First 10 coefficients:', model_l2.coef_[0][:10])

print('Accuracy:', accuracy_score(y_test, model_l2.predict(X_test)))
```



```
First 10 coefficients: [ 2.13248406e+00  1.52771940e-01 -1.45091255e-01 -8.28669349e-04
 -1.42636015e-01 -4.15568847e-01 -6.51940282e-01 -3.44456106e-01
 -2.07613380e-01 -2.97739324e-02]
Accuracy: 0.956140350877193
```

Question 7: Python program to train a Logistic Regression model for multiclass classification using multi_class=”ovr” and print the classification report.

Answer:

```
# Multiclass classification with multi_class='ovr' on iris

from sklearn.datasets import load_iris

from sklearn.linear_model import LogisticRegression

from sklearn.metrics import classification_report

from sklearn.model_selection import train_test_split
```

iris = load_iris(as_frame=True)

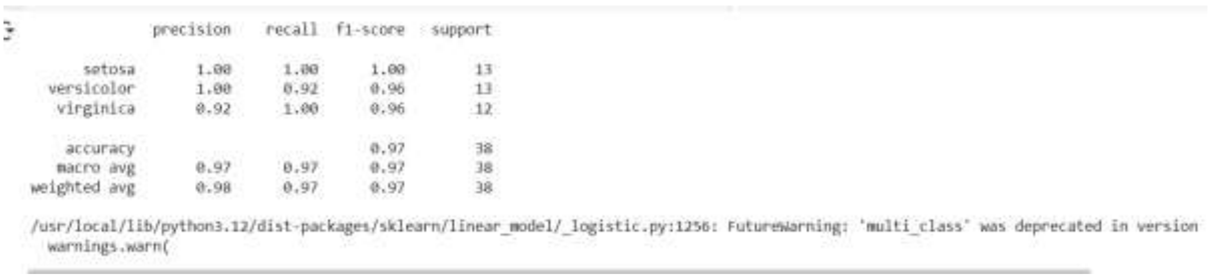
X = iris.data; y = iris.target

Xtr, Xte, ytr, yte = train_test_split(X, y, test_size=0.25, random_state=0, stratify=y)

model_ovr = LogisticRegression(multi_class='ovr', solver='liblinear', max_iter=1000)

model_ovr.fit(Xtr, ytr)

print(classification_report(yte, model_ovr.predict(Xte), target_names=iris.target_names))



```
              precision    recall  f1-score   support

      setosa       1.00      1.00      1.00        13
  versicolor       1.00      0.92      0.96        13
   virginica       0.92      1.00      0.96        12

    accuracy                           0.97        38
   macro avg       0.97      0.97      0.97        38
weighted avg       0.98      0.97      0.97        38

/usr/local/lib/python3.12/dist-packages/sklearn/linear_model/_logistic.py:1256: FutureWarning: 'multi_class' was deprecated in version
  warnings.warn(
```

Question 8: Python program to apply GridSearchCV to tune C and penalty hyperparameters for Logistic Regression and print the best parameters and validation accuracy.

Answer:

# GridSearchCV tuning C and penalty

from sklearn.model_selection import GridSearchCV

from sklearn.linear_model import LogisticRegression

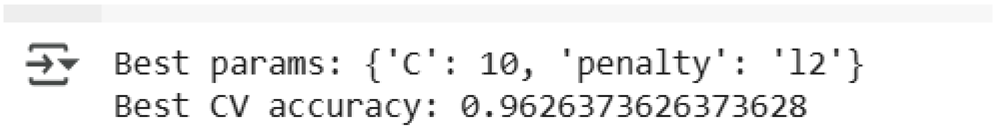
param_grid = {'C': [0.01, 0.1, 1, 10], 'penalty': ['l1', 'l2']}

grid = GridSearchCV(LogisticRegression(solver='liblinear', max_iter=1000), param_grid, cv=5, scoring='accuracy')

grid.fit(X_train, y_train)

print('Best params:', grid.best_params_)

print('Best CV accuracy:', grid.best_score_)

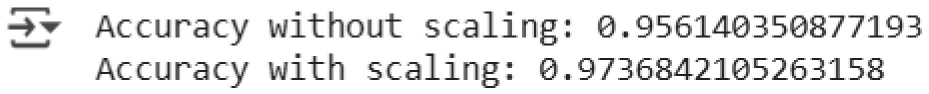


```
Best params: {'C': 10, 'penalty': 'l2'}
Best CV accuracy: 0.9626373626373628
```

Question 9: Python program to standardize the features before training Logistic Regression and compare the model's accuracy with and without scaling .

Answer:

```
# Compare with and without scaling

from sklearn.preprocessing import StandardScaler

from sklearn.linear_model import LogisticRegression

from sklearn.metrics import accuracy_score


scaler = StandardScaler()

X_train_s = scaler.fit_transform(X_train)

X_test_s = scaler.transform(X_test)

model_no_scale = LogisticRegression(solver='liblinear', max_iter=1000).fit(X_train, y_train)

model_scaled = LogisticRegression(solver='liblinear', max_iter=1000).fit(X_train_s, y_train)

print('Accuracy without scaling:', accuracy_score(y_test, model_no_scale.predict(X_test)))

print('Accuracy with scaling:', accuracy_score(y_test, model_scaled.predict(X_test_s)))
```



```
Accuracy without scaling: 0.956140350877193
Accuracy with scaling: 0.9736842105263158
```

Note: Scaling often improves optimization and sometimes model performance, especially when regularization is used or when features are on very different scales.

Question 10: Real- world approach for imbalanced dataset (5% responders) to build a Logistic Regression model for predicting campaign response

Answer: When the positive class (responders) is only 5% , special care is required.

Here's a step-by-step approach:

1. Understand business objective & costs
    - Clarify the relative cost of false positives (wasting marketing spend) vs false negatives (missed revenue). This guides metric selection and thresholding.
2. Data handling & exploration
    - Explore feature distributions, missing values, outliers and correlation.
    - Create domain-derived features (recency/ frequency/monetary (RFM) for e-commerce), interaction terms, and temporal features.
3. Feature preprocessing and scaling

- Impute missing values appropriately (median for skewed numeric, mode for categorical, or model-based imputation).
- Encode categorical variables (one-hot for low cardinality, target encoding or ordinal encoding for high-cardinality categories, with cross – validation to avoid leakage).
- Standardize/normalize numeric features if using regularization or gradiant-based solvers.

4. Address class imbalance Option (often combined for best effect):
    - Resampling:
    - Oversampling (e,g., SMOTE variants)