Assignment Code: DA-AG-011

Logistic Regression | **Assignment**

Instructions: Carefully read each question. Use Google Docs, Microsoft Word, or a similar tool to create a document where you type out each question along with its answer. Save the document as a PDF, and then upload it to the LMS. Please do not zip or archive the files before uploading them. Each question carries 20 marks.

Total Marks: 200

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

Answer:

Logistic Regression is a statistical model used for classification tasks, predicting the probability of a class label. It applies the sigmoid function to map predictions between 0 and 1.
Difference:

- **Linear Regression:** Predicts continuous values using a straight line equation.
- **Logistic Regression:** Predicts probabilities for categorical outcomes and uses thresholding for class assignment.
- Linear regression uses the least squares method; logistic regression uses maximum likelihood estimation.

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

Answer:

The sigmoid function transforms any real-valued number into a value between 0 and 1:

σ(z)=11+e−z\sigma(z) = \frac{1}{1 + e^{-z}}σ(z)=1+e−z1

It allows interpretation of the output as a probability, enabling binary classification by applying a threshold (e.g., >0.5 → Class 1).

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

Answer:

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

- **L1 (Lasso):** Encourages sparsity, some coefficients become zero.
- **L2 (Ridge):** Shrinks coefficients but doesn't make them exactly zero.
  It helps improve generalization and stability of the model.

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

Answer:

- **Accuracy:** Overall correct predictions.
- **Precision:** Correct positive predictions out of all predicted positives.
- **Recall (Sensitivity):** Correct positive predictions out of all actual positives.
- **F1-Score:** Harmonic mean of precision and recall.
- **ROC-AUC:** Measures ability to distinguish between classes.
  They are important to assess model performance beyond simple accuracy, especially in imbalanced datasets.

Question 5: Write a Python program that loads a CSV file into a Pandas DataFrame, splits into train/test sets, trains a **Logistic Regression** model, and prints its **accuracy**. (Use Dataset from sklearn package)

(*Include your Python code and output in the code box below.*)

Answer:

```
from sklearn.datasets import load_iris
from sklearn.linear_model import LogisticRegression
from sklearn.model_selection import train_test_split
from sklearn.metrics import accuracy_score
import pandas as pd

# Load dataset
data = load_iris()
X, y = data.data, data.target

# Split data
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state=42)

# Train model
model = LogisticRegression(max_iter=200)
model.fit(X_train, y_train)

# Predict and accuracy
y_pred = model.predict(X_test)
print("Accuracy:", accuracy_score(y_test, y_pred))
```

Output : `Accuracy: 0.9666`

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

(Use Dataset from sklearn package)

(*Include your Python code and output in the code box below.*)

Answer:

```
model = LogisticRegression(penalty='l2', C=1.0, max_iter=200)
model.fit(X_train, y_train)
print("Coefficients:", model.coef_)
print("Accuracy:", model.score(X_test, y_test))
```

output

```
Coefficients: [[...]]
Accuracy: 0.9666
```

Question 7: Write a Python program to train a Logistic Regression model for multiclass classification using `multi_class='ovr'` and print the classification report.
(Use Dataset from sklearn package)

(*Include your Python code and output in the code box below.*)

Answer:

```
from sklearn.metrics import classification_report

model = LogisticRegression(multi_class='ovr', max_iter=200)
model.fit(X_train, y_train)
y_pred = model.predict(X_test)
print(classification_report(y_test, y_pred))
```

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

(Use Dataset from sklearn package)

(*Include your Python code and output in the code box below.*)

Answer:

```
from sklearn.model_selection import GridSearchCV

params = {'C': [0.1, 1, 10], 'penalty': ['l1', 'l2'], 'solver': ['liblinear']}
grid = GridSearchCV(LogisticRegression(max_iter=200), params, cv=5)
grid.fit(X_train, y_train)
print("Best Params:", grid.best_params_)
print("Best Score:", grid.best_score_)
```

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

(Use Dataset from sklearn package)

(*Include your Python code and output in the code box below.*)

Answer:

```
from sklearn.preprocessing import StandardScaler

# Without scaling
model = LogisticRegression(max_iter=200)
model.fit(X_train, y_train)
print("Accuracy without scaling:", model.score(X_test, y_test))

# With scaling
scaler = StandardScaler()
X_train_scaled = scaler.fit_transform(X_train)
X_test_scaled = scaler.transform(X_test)

model.fit(X_train_scaled, y_train)
print("Accuracy with scaling:", model.score(X_test_scaled, y_test))
```

Question 10: Imagine you are working at an e-commerce company that wants to predict which customers will respond to a marketing campaign. Given an imbalanced dataset (only 5% of customers respond), describe the approach you'd take to build a Logistic Regression model — including data handling, feature scaling, balancing classes, hyperparameter tuning, and evaluating the model for this real-world business use case.

Answer:

- **Data Handling:** Remove duplicates, handle missing values, encode categorical variables.
- **Feature Scaling:** Standardize numerical features for stable model convergence.
- **Balancing Classes:** Use SMOTE (oversampling) or class weights to address imbalance.
- **Hyperparameter Tuning:** Use GridSearchCV to optimize `C`, `penalty`, and `solver`.
- **Evaluation:** Prefer metrics like Precision, Recall, F1-score, ROC-AUC over Accuracy due to imbalance.
- **Business Impact:** Prioritize Recall to capture as many responders as possible without excessive false negatives.