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 supervised machine learning algorithm used for classification problems. Unlike linear regression which predicts continuous values it predicts the probability that an input belongs to a specific class. It is used for binary classification where the output can be one of two possible categories such as Yes/No, True/False or 0/1. It uses sigmoid function to convert inputs into a probability value between 0 and 1.

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

Answer:

Understanding Sigmoid Function

1. The sigmoid function is an important part of logistic regression, which is used to convert the raw output of the model into a probability value between 0 and 1.
2. This function takes any real number and maps it into the range 0 to 1, forming an "S"-shaped curve called the sigmoid curve or logistic curve. Because probabilities must lie between 0 and 1, the sigmoid function is perfect for this purpose.
3. In logistic regression, we use a threshold value, usually 0.5, to decide the class label.

If the sigmoid output is same or above the threshold, the input is classified as Class 1. If it is below the threshold, the input is classified as Class 0. This approach helps to transform continuous input values into meaningful class predictions.

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

Answer:

Regularization is an important technique in machine learning that helps to improve model accuracy by preventing overfitting which happens when a model learns the training data too well including noise and outliers and perform poor on new data.
By adding a penalty for complexity it helps simpler models to perform better on new data.

The various benefits of regularization are as follows:

1. Prevents Overfitting: Regularization helps models focus on underlying patterns instead of memorizing noise in the training data.
2. Improves Interpretability: L1 (Lasso) regularization simplifies models by reducing less important feature coefficients to zero.
3. Enhances Performance: Prevents excessive weighting of outliers or irrelevant features, which helps in improving overall model accuracy.

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

Answer:

Classification is a supervised machine-learning technique that predicts the class label based on the input data. There are different classification algorithms to build a classification model, such as Stochastic Gradient Classifier, Support Vector Machine Classifier, Random Forest Classifier, etc. To choose the right model, it is important to gauge the performance of each classification algorithm. Understanding classification evaluation metrics is crucial for assessing the performance of machine learning models, especially in tasks like binary or multiclass classification.
Some common metrics are:

- Accuracy
- Confusion Matrix
- Precision, Recall and F1 Score
- AUC-ROC Curve

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:

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

data = load_iris()
X = pd.DataFrame(data.data, columns = data.feature_names)
y = data.target

X_train, X_test, y_train, y_test = train_test_split(X, y, test_size = 0.3, random_state = 1)

model = LogisticRegression(max_iter = 200)
model.fit(X_train, y_train)
y_pred = model.predict(X_test)

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

Output:
Model Accuracy: 0.98
```

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:

```
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

data = load_breast_cancer()
X = data.data
y = data.target

X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.3, random_state=1)

model = LogisticRegression(penalty='l2', C=0.1, solver='liblinear', max_iter=1000)
model.fit(X_train, y_train)

print("Model Coefficients:")
print(model.coef_[0])

y_pred = model.predict(X_test)
accuracy = accuracy_score(y_test, y_pred)
print(f"\nModel Accuracy: {accuracy:.4f}")


Output:

Model Coefficients:
[ 0.50770704  0.10663658  0.37305199 -0.01773295 -0.01376922 -0.08048916
 -0.11403382 -0.04816543 -0.02442704 -0.00240325  0.02482266  0.12836137
  0.0194768  -0.04815176 -0.00167733 -0.01710842 -0.02647023 -0.00729868
 -0.00423199 -0.00127577  0.57300044 -0.20903336 -0.23212739 -0.01220784
 -0.02615327 -0.24438662 -0.31372616 -0.09772481 -0.05715664 -0.01799687]

Model Accuracy: 0.9415
```

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.datasets import load_iris
from sklearn.model_selection import train_test_split
from sklearn.linear_model import LogisticRegression
```

```
from sklearn.metrics import classification_report

data = load_iris()

X, y = data.data, data.target

X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.3, random_state=1)

model = LogisticRegression(multi_class='ovr', solver='liblinear', random_state=42)
model.fit(X_train, y_train)

y_pred = model.predict(X_test)

print("Classification Report:")
print(classification_report(y_test, y_pred, target_names=data.target_names))


Output:

Classification Report:
              precision    recall  f1-score   support

      setosa       1.00      1.00      1.00        14
  versicolor       1.00      0.72      0.84        18
   virginica       0.72      1.00      0.84        13

    accuracy                           0.89        45
   macro avg       0.91      0.91      0.89        45
weighted avg       0.92      0.89      0.89        45
```

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.datasets import load_breast_cancer
from sklearn.linear_model import LogisticRegression
```

```python
from sklearn.model_selection import GridSearchCV, train_test_split

data = load_breast_cancer()
X,y = data.data, data.target

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

param_grid = {
    'C':[0.001, 0.01, 0.1, 1, 10, 100],
    'penalty':['l1','l2']
}

logistic_model = LogisticRegression(solver='liblinear', max_iter=1000)

grid_search = GridSearchCV(logistic_model, param_grid, cv=5,
scoring='accuracy')

grid_search.fit(X_train, y_train)

print("Best Parameters:", grid_search.best_params_)

print("Best Validation Accuracy:", grid_search.best_score_)

#Evaluate the best model on the test set(optional, but good practice)
test_accuracy = grid_search.score(X_test, y_test)
print("Test Accuracy with Best Model:", test_accuracy)
```

Output:

```
Best Parameters: {'C': 100, 'penalty': 'l1'}
Best Validation Accuracy: 0.9670329670329672
Test Accuracy with Best Model: 0.9824561403508771
```

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

data = load_breast_cancer()
X = data.data
y = data.target

X_train, X_test, y_train, y_test = train_test_split(X, y, test_size = 0.3,
random_state = 42)

print("--- Model without scaling ---")
logistic_regression_unscaled = LogisticRegression(max_iter=5000,
random_state=42)
logistic_regression_unscaled.fit(X_train, y_train)
y_pred_unscaled = logistic_regression_unscaled.predict(X_test)
accuracy_unscaled = accuracy_score(y_test, y_pred_unscaled)
print(f"Accuracy without scaling: {accuracy_unscaled:.4f}")

print("\n--- Model with scaling ---")
scaler = StandardScaler()
X_train_scaled = scaler.fit_transform(X_train)
X_test_scaled = scaler.transform(X_test)

logistic_regression_scaled = LogisticRegression(max_iter=1000,
random_state=42)
logistic_regression_scaled.fit(X_train_scaled, y_train)
y_pred_scaled = logistic_regression_scaled.predict(X_test_scaled)
accuracy_scaled = accuracy_score(y_test, y_pred_scaled)
print(f"Accuracy with scaling: {accuracy_scaled:.4f}")

print(f"\nAccuracy difference (scaled - unscaled): {accuracy_scaled -
accuracy_unscaled:.4f}")
```

Output:

```
--- Model without scaling ---
Accuracy without scaling: 0.9766

--- Model with scaling ---
Accuracy with scaling: 0.9825

Accuracy difference (scaled - unscaled): 0.0058
```

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:

```
**Techniques to Handle Imbalanced Data Set Problem**

In rare cases like fraud detection or disease prediction, it is vital to
identify the minority classes correctly. So, the model should not be
biased to detect only the majority class but should give equal weight or
importance to the minority class, too. Here, I discuss some techniques to
handle imbalanced dataset problem. There is no correct or wrong method;
different techniques work well with other problems.

1. Data Prep & Scaling

 a) Handle missing values, encode categoricals (one-hot), and scale
numericals with StandardScaler (Logistic Regression is sensitive to
feature scale).

2. Handle Imbalance

 a) Use SMOTE (on training set only) to oversample the 5% responder class.

 b) Try class_weight='balanced' as an alternative.
```

```
3. Model Building

 a) Create a pipeline: Scaling → SMOTE → Logistic Regression.

 b) Tune hyperparameters (C, penalty, solver, class_weight) via
GridSearchCV with stratified folds.

4. Threshold Optimization

 a) Get probabilities (predict_proba) and choose optimal threshold using
Precision-Recall curve to maximize recall without killing precision.

5. Evaluation

 a) Use F1-score (primary), ROC-AUC, and PR-AUC—not accuracy.

 b) From a business view, check ROI of predicted responders.
```