Assignment Code: DA-AG-011

Logistic Regression | **Assignment**

Name: Dibyojyoti Dutta
Email-Id:- dduttaoffice184@gmail.com
Assignment Name:- Logistic Regression
Drive Link:-N/A
Github Link:- Link

Total Marks: 200

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

Answer:

Logistic regression is a statistical and machine learning method used to predict whether something belongs to a category — most often for yes/no, true/false, or 0/1 outcomes.

Logistic Regression and Linear Regression are both popular statistical methods for modeling relationships between variables, but Instead of giving a number (like linear regression does), Logistic regression gives a probability between 0 and 1, which you can then convert into a category.

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

Answer:

The sigmoid function plays a central role in logistic regression by converting any real-valued input into a value between 0 and 1, making it possible to interpret the output as a probability.

- Steps:- Logistic regression first calculates a linear combination of the input features (similar to what is done in linear regression).
- This linear result (which could be any value from minus infinity to plus infinity) is then passed through the sigmoid function.
- The sigmoid function “squashes” the linear output into the range (0, 1), turning it into something that can be treated as a probability of belonging to the "1" (yes/positive) class

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

Answer:

Regularization helps by adding a penalty term to the loss function, which restricts the growth of coefficients and makes the model simpler and more generalizable.

- The regularization penalty discourages the model from assigning too high values to any coefficient.
- Common types of regularization:
  - L1 regularization (Lasso): Adds the absolute values of the coefficients as a penalty. It can shrink some coefficients exactly to zero, effectively performing feature selection.
  - L2 regularization (Ridge): Adds the squared values of the coefficients as a penalty. It shrinks coefficients toward zero but does not eliminate any.
- This penalty is controlled by a hyperparameter (often called lambda or α). A higher value means stronger regularization.

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

Answer:

1. Accuracy → % of predictions that are correct. *(Good for balanced data)*
2. Precision → Of all predicted positives, how many were actually correct? *(Avoid false alarms)*
3. Recall (Sensitivity) → Of all real positives, how many did we find? *(Avoid missing cases)*
4. F1 Score → Balance between precision & recall. *(Good when both matter)*
5. AUC-ROC → How well the model separates classes overall. *(Higher = better)*

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
import numpy as np
import seaborn as sns
import matplotlib.pyplot as plt
```

```
import warnings
warnings.filterwarnings('ignore')

from sklearn.datasets import load_iris
data= load_iris()
#data
df=pd.DataFrame(data.data, columns=data.feature_names)
df['target'] = data.target
df

X=df.drop('target',axis=1)
y=df['target']

from sklearn.model_selection import train_test_split
X_train,X_test,y_train,y_test=train_test_split(X,y,test_size=0.25,random_stat
e=1)

from sklearn.linear_model import LogisticRegression
model=LogisticRegression()
model

model.fit(X_train,y_train)
```

```
y_pred=model.predict(X_test)
y_pred
```

```
from sklearn.metrics import
accuracy_score,confusion_matrix,classification_report
print("Accuracy: ",accuracy_score(y_test,y_pred))
```

Output:-

```
Accuracy:  0.9736842105263158
```

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_iris
data = load_iris()

df = pd.DataFrame(data.data, columns=data.feature_names)
df['target'] = data.target
df

X = df.drop('target', axis=1)
y = df['target']

from sklearn.model_selection import train_test_split
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.25,
random_state=1)

# L2 regularization is default
from sklearn.linear_model import LogisticRegression
model = LogisticRegression()
model

model.fit(X_train, y_train)

y_pred = model.predict(X_test)
y_pred
model = LogisticRegression(penalty='l2', max_iter=200)


from sklearn.metrics import accuracy_score, confusion_matrix,
classification_report
print("Accuracy: ", accuracy_score(y_test, y_pred))

model.fit(X_train, y_train)
print("\nModel Coefficients:")
print(pd.DataFrame(model.coef_, columns=data.feature_names))
```

Output:-

```
Accuracy:  0.9736842105263158


Model Coefficients:
   sepal length (cm)  sepal width (cm)  petal length (cm)  petal width (cm)
0          -0.430520          0.804470          -2.304520         -0.950636
1           0.629204         -0.424674          -0.206373         -0.783506
2          -0.198684         -0.379796           2.510893          1.734142
```

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
data = load_iris()

df = pd.DataFrame(data.data, columns=data.feature_names)
df['target'] = data.target
df

X = df.drop('target', axis=1)
y = df['target']

from sklearn.model_selection import train_test_split
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.25,
random_state=1)

model = LogisticRegression(multi_class='ovr', max_iter=200)

model.fit(X_train, y_train)

y_pred = model.predict(X_test)

print("Classification Report:\n")
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        13
  versicolor       1.00      0.75      0.86        16
   virginica       0.69      1.00      0.82         9

    accuracy                           0.89        38
   macro avg       0.90      0.92      0.89        38

weighted avg       0.93      0.89      0.90        38
```

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_iris
data = load_iris()

df = pd.DataFrame(data.data, columns=data.feature_names)
df['target'] = data.target
df

X = df.drop('target', axis=1)
y = df['target']

from sklearn.model_selection import train_test_split, GridSearchCV
from sklearn.linear_model import LogisticRegression

X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.25,
random_state=1)

model = LogisticRegression(max_iter=1000, solver='saga')

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

```
grid = GridSearchCV(model, param_grid, cv=5)
grid.fit(X, y)

print("Best parameters:", grid.best_params_)
print("Validation accuracy: {:.4f}".format(grid.best_score_))
```

Output:-

```
Best parameters: {'C': 1, 'penalty': 'l1'}
Validation accuracy: 0.9800
```

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:

```
import warnings
warnings.filterwarnings('ignore')

from sklearn.datasets import load_iris
data = load_iris()

df = pd.DataFrame(data.data, columns=data.feature_names)
df['target'] = data.target
df

X = df.drop('target', axis=1)
y = df['target']

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

X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.25,
random_state=1)

# 1. Logistic Regression without scaling
model_no_scaling = LogisticRegression(max_iter=1000, solver='saga')
model_no_scaling.fit(X_train, y_train)
y_pred_no_scaling = model_no_scaling.predict(X_test)
acc_no_scaling = accuracy_score(y_test, y_pred_no_scaling)

# 2. Logistic Regression with StandardScaler
scaler = StandardScaler()
```

```
X_train_scaled = scaler.fit_transform(X_train)
X_test_scaled = scaler.transform(X_test)

model_with_scaling = LogisticRegression(max_iter=1000, solver='saga')
model_with_scaling.fit(X_train_scaled, y_train)
y_pred_with_scaling = model_with_scaling.predict(X_test_scaled)
acc_with_scaling = accuracy_score(y_test, y_pred_with_scaling)

print(f"Accuracy without scaling: {acc_no_scaling:.4f}")
print(f"Accuracy with scaling: {acc_with_scaling:.4f}")
```

Output:-

```
Accuracy without scaling: 0.9737
Accuracy with scaling: 0.97
```

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: Approach:-

1. Data Handling and Preprocessing

- Load and explore data: Check class distribution and missing values.
- Train-test split: Use stratified split to maintain imbalance ratio in training and test sets.

2. Feature Scaling

- Use StandardScaler to standardize features (mean=0, std=1) before training logistic regression.
- Scaling helps convergence and model performance as logistic regression is sensitive to feature scales.

3. Addressing Class Imbalance

- Since responders are only 5%, the dataset is highly imbalanced.
- To handle this:
  - Class weighting: Use class_weight='balanced' in LogisticRegression to assign higher weight to minority class.
  - Or use sample weighting: Provide weights explicitly for each sample during training.
  - Alternative data-level approaches:
    - Oversampling minority class using techniques like SMOTE to synthetically generate minority samples.
    - Undersampling majority class to reduce its dominance.
  - Weighting or resampling can be combined or used based on experimentation.

4. Model Building and Hyperparameter Tuning

- Use Logistic Regression with solver='saga' (works for both L1 and L2 penalties).
- Hyperparameters to tune with GridSearchCV or RandomizedSearchCV include:
  - Regularization strength C (e.g., [0.01, 0.1, 1, 10])
  - Penalty type: 'l1' or 'l2'
  - Class weights (if tuning this parameter)
- Incorporate feature scaling in a pipeline with LogisticRegression.

5. Model Evaluation Metrics (with Imbalance Consideration)

- Accuracy can be misleading due to imbalance.
- Use metrics that better reflect minority class performance:
    - Precision, Recall, F1-score (especially F1 or F2 to emphasize recall)
    - Area Under ROC Curve (AUC-ROC)
    - Area Under Precision-Recall Curve (AUC-PR) — useful for imbalanced data.
- Use confusion matrix analysis to understand false positives/negatives impact.