NAME:- JOBA ADHIKARY

EMAIL ID:- joba.adhikary55@gmail.com

#1. What is Logistic Regression, and how does it differ from Linear Regression?

--Logistic Regression is a statistical method used to model the probability of a binary outcome based on one or more independent variables.

The differences between Logistic and Linear Regression are:-

*Logistic Regression:-

>>It's purpose is to Predicts a categorical dependent variable

>>It's output range is between 0and 1

>>It assumes a linear relationship between independent variables and the log-odds of the dependent variable.

*Linear Regression:-

>>It's purose is to Predict a continous dependent variable.

>>It's output range is any real number.

>> It assumes a linear relationship between independent and dependent variables.

#2. Explain the role of the Sigmoid function in Logistic Regression.

--In Logistic Regression, the sigmoid function plays a crucial role in transforming the output of the model into a probability between 0 and 1.

- The sigmoid curve is S-shaped curve, with a smooth transition from 0 to 1.
- This allows small changes in input to have big effects near the center but minimal changes near the extremes

#3. What is Regularization in Logistic Regression and why is it needed?

--Regularization in Logistic Regression is a technique used to prevent overfitting by adding a penalty term to the loss function, which discourages the model from assigning excessively large weights to features.

It is needed because:-

- Without regularization, the model might learn overly complex patterns that fit the training data very well but fail on unseen data.
- There are too many features relative to the number of observation.

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

--Some common evaluations are:-

- Precision
- Accuracy
- Recall
- F1 score
- Log Loss
- ROC-AUC
- Confusion Matrix

#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

--

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

```
-- import pandas as pd

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
data = {
    'Feature1': [2.5, 1.5, 3.6, 4.0, 2.8, 3.0, 3.8, 4.2],
    'Feature2': [1.2, 3.4, 2.1, 3.0, 2.8, 3.2, 3.9, 4.4],
    'Target':    [0, 0, 1, 1, 0, 1, 1, 1]
}
df = pd.DataFrame(data)
x = df[['Feature1', 'Feature2']]
y = df['Target']
X_train, X_test, y_train, y_test = train_test_split(
    X, y, test_size=0.25, random_state=42
)
scaler = StandardScaler()
X_train = scaler.fit_transform(X_train)
X_test = scaler.transform(X_test)
y_pred = model.predict(X_test)
accuracy = accuracy_score(y_test, y_pred)
print("Model Coefficients:", model.coef_)
print("Model Intercept:", model.intercept_)
```

```
print(f"Accuracy: {accuracy:.2f}")
```

>>OUTPUT:-

```
Model Coefficients: [[1.09320679 0.36792983]]
Model Intercept: [0.93716805]

Accuracy: 1.00
```

#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)

```
-- import numpy as np

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
iris = load_iris()
X = iris.data
y = iris.target
X_train, X_test, y_train, y_test = train_test_split(
    X, y, test_size=0.2, random_state=42
)
model = LogisticRegression(multi_class='ovr', solver='lbfgs',
max_iter=1000)
model.fit(X_train, y_train)
y_pred = model.predict(X_test)
print("Classification Report:\n", classification_report(y_test, y_pred,
target_names=iris.target_names))
```

>>OUTPUT:-

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

      setosa       1.00      1.00      1.00        10
  versicolor       1.00      0.89      0.94         9
   virginica       0.92      1.00      0.96        11

    accuracy                           0.97        30
   macro avg       0.97      0.96      0.97        30
weighted avg       0.97      0.97      0.97        30

/usr/local/lib/python3.11/dist-
packages/sklearn/linear_model/_logistic.py:1256: FutureWarning: 'multi_class'
was deprecated in version 1.5 and will be removed in 1.7. Use
```

```
OneVsRestClassifier(LogisticRegression(..)) instead. Leave it to its default
value to avoid this warning.
```

#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

```
-- import numpy as np

from sklearn.datasets import load_iris
from sklearn.model_selection import train_test_split, GridSearchCV
from sklearn.linear_model import LogisticRegression
iris = load_iris()
X = iris.data
y = iris.target
X_train, X_test, y_train, y_test = train_test_split(
    X, y, test_size=0.2, random_state=12
)
log_reg = LogisticRegression(max_iter=10, solver='liblinear',
multi_class='ovr')
param_grid ={
    'C': [0.01, 0.1, 1, 10, 12],
    'penalty': ['1', 'l']
}
grid_search = GridSearchCV(log_reg, param_grid, cv=5, scoring='accuracy')
grid_search.fit(X_train, y_train)
print("Best Parameters:", grid_search.best_params_)
print(f"Best Cross-Validation Accuracy: {grid_search.best_score_:.4f}")
```

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

```
-- import numpy as np

from sklearn.datasets import load_iris
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
iris = load_iris()
X = iris.data
y = iris.target
X_train, X_test, y_train, y_test = train_test_split(
    X, y, test_size=0.2, random_state=12
)
```

```
model_no_scaling = LogisticRegression(max_iter=1000, solver='lbfgs',
multi_class='ovr')
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)
scaler = StandardScaler()
X_train_scaled = scaler.fit_transform(X_train)
X_test_scaled = scaler.transform(X_test)
model_scaling = LogisticRegression(max_iter=1000, solver='lbfgs',
multi_class='ovr')
model_scaling.fit(X_train_scaled, y_train)
y_pred_scaling = model_scaling.predict(X_test_scaled)
acc_scaling = accuracy_score(y_test, y_pred_scaling)
print(f"Accuracy without Scaling: {acc_no_scaling:.4f}")
print(f"Accuracy with Scaling: {acc_scaling:.4f}")
```

>>OUTPUT:-

```
Accuracy without Scaling: 0.9667
Accuracy with Scaling: 0.9333
/usr/local/lib/python3.11/dist-
packages/sklearn/linear_model/_logistic.py:1256: FutureWarning: 'multi_class'
was deprecated in version 1.5 and will be removed in 1.7. Use
OneVsRestClassifier(LogisticRegression(..)) instead. Leave it to its default
value to avoid this warning.
  warnings.warn(
/usr/local/lib/python3.11/dist-
packages/sklearn/linear_model/_logistic.py:1256: FutureWarning: 'multi_class'
was deprecated in version 1.5 and will be removed in 1.7. Use
OneVsRestClassifier(LogisticRegression(..)) instead. Leave it to its default
value to avoid this warning.
warnings.warn
```

#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.**

--*STEPS:-

- Clean & engineer data
- Tune hyperparameters
- Handles imbalance
- Evaluate model

*Techniques:-

- One-hot encoding, imputation
- GridSearchCV on C penalty
- Class weight = balanced or SMOTE
- It is based on cost benefit

*Reasons:-

- Handle missing categorical values
- Optimize generalization
- Prevents majority dominance
- Account for imbalance
- Match marketing ROI goals