# Logistic Regression Answer

1. What is Logistic Regression, and how does it differ from Linear Regression?
   - **Logistic Regression** is a statistical and machine learning model used for **classification problems**, especially when the output (dependent variable) is **categorical** — most often **binary** (e.g., Yes/No, Spam/Not Spam, 0/1).
   - Instead of predicting a continuous value like linear regression, logistic regression predicts the **probability** that a given input belongs to a certain class.

- Key Differences: Logistic vs. Linear Regression

| Aspect | Linear Regression | Logistic Regression |
|---|---|---|
| **Purpose** | Predicts continuous numerical values | Predicts probabilities for classification |
| **Output Range** | (−∞,+∞)(-\infty, +\infty) | 00 to 11 (probabilities) |
| **Activation Function** | None (direct output) | Sigmoid (logistic) or other link functions |
| **Loss Function** | Mean Squared Error (MSE) | Log Loss / Binary Cross-Entropy |
| **Best Use Case** | Regression problems (e.g., predicting price) | Classification problems (e.g., spam detection) |
| **Assumption on Residuals** | Errors are normally distributed | Follows Bernoulli distribution |

2. Explain the role of the Sigmoid function in Logistic Regression.
   - The **role of the Sigmoid function in Logistic Regression** is to **convert the raw linear output into a probability between 0 and 1**.

### ● Step-by-step role in Logistic Regression:

1. **Compute a Linear Score**
   Logistic regression first calculates:
   z=b0+b1x1+b2x2+⋯+bnxnz = b_0 + b_1x_1 + b_2x_2 + \dots + b_nx_nz=b0+b1x1+b2x2+⋯+bnxn
   Here, zzz can be any real number (negative, zero, or positive).

2. **Transform to Probability**
   The **sigmoid function**:
   σ(z)=11+e−z\sigma(z) = \frac{1}{1 + e^{-z}}σ(z)=1+e−z1
   maps zzz into the range (0,1)(0,1)(0,1), making it interpretable as a probability.

3. **Decision Making**
 Once we have a probability:

    - If p≥0.5p \ge 0.5p≥0.5 → predict **class 1**

    - If p<0.5p < 0.5p<0.5 → predict **class 0**

4. **Smooth and Differentiable**
 The sigmoid function is smooth and differentiable, which allows **gradient descent** to optimize the model efficiently.

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 to the model's loss function for having large coefficient values (weights).

## Why It's Needed

- In logistic regression, if the model learns **very large weights**, it can become **too sensitive** to small changes in input data.

- This leads to **overfitting** — the model performs well on training data but poorly on unseen data.

- Regularization **controls model complexity**, encouraging simpler models that generalize better.

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

- **classification models** (like logistic regression, decision trees, etc.), we need metrics that tell us **how well the model predicts classes** — not just overall accuracy. Different metrics are important because classification tasks often have **imbalanced classes** or different costs for false positives vs. false negatives.

  Common Evaluation Metrics

### 1. Accuracy

Accuracy=Number of Correct PredictionsTotal Predictions\text{Accuracy} = \frac{\text{Number of Correct Predictions}}{\text{Total Predictions}}Accuracy=Total PredictionsNumber of Correct Predictions

- **Meaning:** Percentage of correctly classified samples.
- **Good when:** Classes are balanced.
- **Limitation:** Misleading if data is imbalanced (e.g., predicting “not spam” for all emails in a 99% non-spam dataset still gives 99% accuracy).

---

### 2. Precision

Precision=True PositivesTrue Positives + False Positives\text{Precision} = \frac{\text{True Positives}}{\text{True Positives + False Positives}}Precision=True Positives  False PositivesTrue Positives

- **Meaning:** Of all predicted positives, how many are actually positive.
- **Good when:** Cost of false positives is high (e.g., predicting cancer when not present).

---

### 3. Recall (Sensitivity / True Positive Rate)

Recall=True PositivesTrue Positives + False Negatives\text{Recall} = \frac{\text{True Positives}}{\text{True Positives + False Negatives}}Recall=True Positives + False NegativesTrue Positives

- **Meaning:** Of all actual positives, how many the model correctly identified.
- **Good when:** Cost of false negatives is high (e.g., missing a fraud transaction).

---

### 4. F1 Score

F1=2×Precision × RecallPrecision + Recall\text{F1} = 2 \times \frac{\text{Precision × Recall}}{\text{Precision + Recall}}F1=2×Precision + RecallPrecision × Recall

- **Meaning:** Harmonic mean of precision and recall.
- **Good when:** You want a balance between precision and recall.

---

### 5. ROC-AUC (Receiver Operating Characteristic – Area Under Curve)

- **Meaning:** Measures the model's ability to distinguish between classes across all thresholds.
- **Good when:** You want a threshold-independent performance measure.
- **AUC close to 1:** Very good separation of classes.

---

### 6. Confusion Matrix

- A table showing **True Positives, True Negatives, False Positives, False Negatives**.
- **Meaning:** Gives detailed insight into exactly where the model is making mistakes.

#### Why They're Important

- **Different problems need different priorities** (e.g., in spam detection, false negatives may be fine, but in cancer detection, they're dangerous).
- **One metric alone can mislead** — especially with imbalanced datasets.
- **Helps choose the right threshold** and model for the business objective.

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)

```
import pandas as pd

from sklearn.datasets import load_Lungs_cancer

from sklearn.model_selection import train_test_split

from sklearn.linear_model import LogisticRegression

from sklearn.metrics import accuracy_score

data = load_Lungs_cancer()
```

```
df = pd.DataFrame(data.data, columns=data.feature_names)

df['target'] = data.target


# Display first 5 rows

print("First 5 rows of the dataset:")

print(df.head())


# Split into features and target

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

y = df['target']

# Split into training and testing sets (80% train, 20% test)

X_train, X_test, y_train, y_test = train_test_split(

    X, y, test_size=0.2, random_state=42

)

# Create and train the Logistic Regression model

model = LogisticRegression(max_iter=10000)  # Increased iterations for convergence

model.fit(X_train, y_train)

# Make predictions on the test set

y_pred = model.predict(X_test)

# Calculate and print accuracy

accuracy = accuracy_score(y_test, y_pred)

print(f"\nLogistic Regression Model Accuracy: {accuracy:.4f}")
```

```
First 5 rows of the dataset:

   mean radius  mean texture  mean perimeter  mean area  mean
smoothness  \
```

```
0        17.99         10.38          122.80     1001.0
0.11840

1        20.57         17.77          132.90     1326.0
0.08474

2        19.69         21.25          130.00     1203.0
0.10960

3        11.42         20.38           77.58      386.1
0.14250

4        20.29         14.34          135.10     1297.0
0.10030


   mean compactness  mean concavity  mean concave points  mean symmetry
\

0           0.27760          0.3001              0.14710         0.2419

1           0.07864          0.0869              0.07017         0.1812

2           0.15990          0.1974              0.12790         0.2069

3           0.28390          0.2414              0.10520         0.2597

4           0.13280          0.1980              0.10430         0.1809


   mean fractal dimension  ...  worst texture  worst perimeter  worst
area  \

0                 0.07871  ...          17.33           184.60
2019.0

1                 0.05667  ...          23.41           158.80
1956.0

2                 0.05999  ...          25.53           152.50
1709.0

3                 0.09744  ...          26.50            98.87
567.7

4                 0.05883  ...          16.67           152.20
1575.0
```

```
   worst smoothness  worst compactness  worst concavity  worst concave
points  \

0            0.1622             0.6656           0.7119
0.2654

1            0.1238             0.1866           0.2416
0.1860

2            0.1444             0.4245           0.4504
0.2430

3            0.2098             0.8663           0.6869
0.2575

4            0.1374             0.2050           0.4000
0.1625


   worst symmetry  worst fractal dimension  target

0          0.4601                  0.11890       0

1          0.2750                  0.08902       0

2          0.3613                  0.08758       0

3          0.6638                  0.17300       0

4          0.2364                  0.07678       0


[5 rows x 31 columns]


Logistic Regression Model Accuracy: 0.9561
```