Question 1:

Logistic Regression is a statistical model used for binary and multiclass classification problems. It predicts the probability of a class by mapping a linear combination of input features through a Sigmoid function,
which outputs values between 0 and 1.

Differences from Linear Regression:
- Purpose: Classification vs Regression
- Function: Sigmoid vs Linear
- Output Range: (0,1) vs (-∞,+∞)
- Error Metric: Log Loss vs MSE

Question 2:

The Sigmoid function maps the linear output of the model into a probability between 0 and 1. Formula: σ(z) = 1 / (1 + e^-z). It allows threshold-based classification decisions.

Question 3:

Regularization prevents overfitting by adding a penalty to large coefficients.
Types:
- L1 (Lasso): Absolute value penalty
- L2 (Ridge): Squared penalty

Question 4:

Common metrics:
- Accuracy
- Precision
- Recall
- F1-Score
- ROC-AUC

They give a complete view of model performance, especially with imbalanced data.

Question 5:

Python Code:

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

data = load_iris()
X = pd.DataFrame(data.data, columns=data.feature_names)
y = pd.Series(data.target)
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state=42)
model = LogisticRegression(max_iter=200)
model.fit(X_train, y_train)
print(f"Accuracy: {model.score(X_test, y_test):.2f}")
```

Question 6:

Python Code:

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

Question 7:

Python Code:

```
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:

Python Code:

```
from sklearn.model_selection import GridSearchCV
param_grid = {'C': [0.01, 0.1, 1, 10], 'penalty': ['l1', 'l2'], 'solver': ['liblinear']}
grid = GridSearchCV(LogisticRegression(max_iter=200), param_grid, cv=5)
grid.fit(X_train, y_train)
print("Best Params:", grid.best_params_)
print("Validation Accuracy:", grid.best_score_)
```

Question 9:

Python Code:

```
from sklearn.preprocessing import StandardScaler
model.fit(X_train, y_train)
print("Accuracy without scaling:", model.score(X_test, y_test))
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:

Approach for Imbalanced Dataset:

1. Data Handling: Remove duplicates, encode categorical variables, handle missing data.
2. Feature Scaling: StandardScaler or MinMaxScaler.
3. Balancing Classes: SMOTE, undersampling, or class weights.
4. Model: Logistic Regression with regularization.
5. Hyperparameter Tuning: GridSearchCV for C, penalty, solver.
6. Evaluation: Precision, Recall, F1, ROC-AUC instead of only accuracy.
7. Deployment: Monitor and retrain periodically.