Logistic Regression (Theoretical and Practical Implementation)

🎓 Part 1: Theoretical Explanation

1. Introduction to Logistic Regression

- **Definition**: Logistic Regression is a statistical model used for binary classification problems. Despite its name, it is used for classification rather than regression tasks.
- **Goal**: Predict the probability that a given input belongs to a particular category (class 0 or class 1).

2. Why Not Linear Regression for Classification?

- Linear regression outputs continuous values, which can exceed the [0,1] range.
- Classification requires probabilistic interpretation.
- Logistic regression addresses this by using the **sigmoid (logistic) function** to constrain output between 0 and 1.

3. The Logistic (Sigmoid) Function

$$\sigma(z) = \frac{1}{1 + e^{-z}}$$

- S-shaped curve.
- Converts linear combination of inputs into a probability.
- Output ranges from 0 to 1.

4. Model Representation

- Input features: $x = (x_1, x_2, \ldots, x_n)$
- Parameters: $\beta = (\beta_0, \beta_1, \ldots, \beta_n)$
- Linear combination: $z = \beta_0 + \beta_1 x_1 + \ldots + \beta_n x_n$
- Prediction: $P(y = 1|x) = \sigma(z) = \frac{1}{1+e^{-z}}$

5. Cost Function: Binary Cross-Entropy (Log-Loss)

$$L(\beta) = -\sum_{i=1}^{m} \left[y_i \log(p_i) + (1 - y_i) \log(1 - p_i) \right]$$

- Measures how well the model's predictions match actual labels.
- Convex, enabling effective optimization using Gradient Descent.

6. Model Optimization

- **Gradient Descent**: Iteratively updates weights to minimize the loss.
- **Regularization**:
  - L1 (Lasso): Promotes sparsity.
  - L2 (Ridge): Penalizes large coefficients to prevent overfitting.

7. Decision Boundary

- Class prediction:

$$y = \begin{cases} 1 & \text{if } P(y = 1|x) \geq 0.5 \\ 0 & \text{otherwise} \end{cases}$$

- Can be visualized in 2D feature space as a line separating classes.

8. Model Assumptions

- Linearity in the log-odds.
- No multicollinearity.
- Independence of observations.

9. ROC Curve (Receiver Operating Characteristic)

- **Definition**: A plot of the True Positive Rate (TPR) against the False Positive Rate (FPR) at various threshold settings.
- **True Positive Rate (Recall)**: $TPR = \frac{TP}{TP+FN}$
- **False Positive Rate**: $FPR = \frac{FP}{FP+TN}$
- Helps visualize the trade-off between sensitivity and specificity.
- **AUC (Area Under Curve)**: Measures the overall ability of the model to distinguish between classes. AUC close to 1.0 indicates a good model.

10. Model Interpretation

- The coefficients β_i indicate the effect of each feature on the log-odds of the outcome.
- To interpret:
  - Convert to **odds ratio**: $OR = e^{\beta_i}$
  - $OR > 1$: Feature increases the odds of the positive class.
  - $OR < 1$: Feature decreases the odds.
- Useful for understanding feature importance and direction of influence.

💪 Part 2: Practical Implementation (Python)

```
# Import Libraries
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns
from sklearn.model_selection import train_test_split
from sklearn.linear_model import LogisticRegression
from sklearn.metrics import classification_report, confusion_matrix, roc_auc_score, roc_curv
```

```
# Load Dataset
from sklearn.datasets import load_breast_cancer

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

	mean radius	mean texture	mean perimeter	mean area	mean smoothness	mean compactness	mean concavity	mean concave points	m symme
0	17.99	10.38	122.80	1001.0	0.11840	0.27760	0.3001	0.14710	0.2
1	20.57	17.77	132.90	1326.0	0.08474	0.07864	0.0869	0.07017	0.1
2	19.69	21.25	130.00	1203.0	0.10960	0.15990	0.1974	0.12790	0.2
3	11.42	20.38	77.58	386.1	0.14250	0.28390	0.2414	0.10520	0.2
4	20.29	14.34	135.10	1297.0	0.10030	0.13280	0.1980	0.10430	0.1

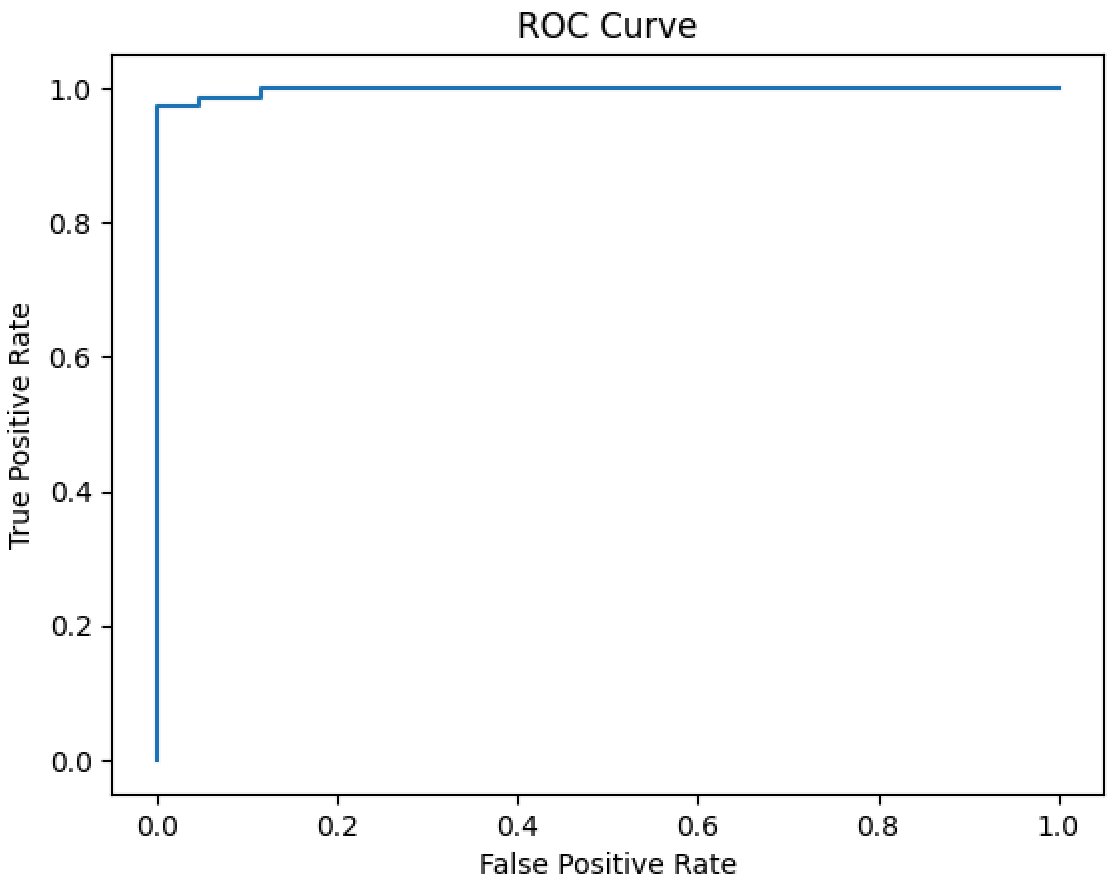
5 rows × 31 columns

```
# Data Preprocessing
df.isnull().sum()

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

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

```
# Model Training and Evaluation
model = LogisticRegression(max_iter=10000)
model.fit(X_train, y_train)

y_pred = model.predict(X_test)
print(confusion_matrix(y_test, y_pred))
print(classification_report(y_test, y_pred))
```

```
[[39  4]
 [ 1 70]]
              precision    recall  f1-score   support

           0       0.97      0.91      0.94        43
           1       0.95      0.99      0.97        71

    accuracy                           0.96       114
   macro avg       0.96      0.95      0.95       114
weighted avg       0.96      0.96      0.96       114
```

```
# ROC Curve
y_proba = model.predict_proba(X_test)[:, 1]
fpr, tpr, thresholds = roc_curve(y_test, y_proba)

plt.plot(fpr, tpr)
plt.xlabel("False Positive Rate")
plt.ylabel("True Positive Rate")
plt.title("ROC Curve")
plt.show()

print("AUC Score:", roc_auc_score(y_test, y_proba))
```




```
AUC Score: 0.9977071732721913
```

```
# Model Interpretation
coeff_df = pd.DataFrame({'Feature': X.columns, 'Coefficient': model.coef_[0]})
coeff_df['Odds Ratio'] = np.exp(coeff_df['Coefficient'])
coeff_df.sort_values(by='Odds Ratio', ascending=False)
```

	Feature	Coefficient	Odds Ratio
11	texture error	1.370567	3.937581
0	mean radius	1.027437	2.793895
1	mean texture	0.221451	1.247885
20	worst radius	0.111653	1.118125
15	compactness error	0.047361	1.048500
3	mean area	0.025467	1.025794
19	fractal dimension error	0.011605	1.011673
22	worst perimeter	-0.015554	0.984566
23	worst area	-0.016857	0.983284
14	smoothness error	-0.022455	0.977795
17	concave points error	-0.032402	0.968117
18	symmetry error	-0.034737	0.965859
9	mean fractal dimension	-0.036494	0.964163
16	concavity error	-0.042948	0.957961
13	area error	-0.087196	0.916498
10	radius error	-0.097102	0.907463
29	worst fractal dimension	-0.100944	0.903984
4	mean smoothness	-0.156235	0.855358
12	perimeter error	-0.181409	0.834094
8	mean symmetry	-0.226682	0.797174
5	mean compactness	-0.237713	0.788429
7	mean concave points	-0.283692	0.752998
24	worst smoothness	-0.307731	0.735113
2	mean perimeter	-0.362135	0.696188
21	worst texture	-0.508877	0.601170
27	worst concave points	-0.510929	0.599938
6	mean concavity	-0.532558	0.587101
28	worst symmetry	-0.746894	0.473836
25	worst compactness	-0.772709	0.461760
26	worst concavity	-1.428595	0.239645

📆 Assignment for Students

Objective: Apply logistic regression to a new dataset.

1. Choose a binary classification dataset from Kaggle or UCI (e.g., Titanic, Pima Indians Diabetes)