Lecture 14: Advanced Logistic Regression and Classification Evaluation

CS109A: Introduction to Data Science

Harvard University

■ **Course:** CS109A: Introduction to Data Science

■ **Lecture:** Lecture 14

■ **Instructors:** Pavlos Protopapas, Kevin Rader, Chris Tanner

■ **Topics:** Logistic Regression Inference, Interactions, Decision Boundaries, Regularization, Multiclass Classification, Confusion Matrix, ROC/AUC

Key Summary

This lecture builds upon the foundations of logistic regression to cover advanced topics essential for practical classification tasks.

Key Topics:

- Inference in logistic regression: confidence intervals and hypothesis tests using Z-statistics
- Interpreting coefficients with binary predictors and the concept of reference groups
- Multiple logistic regression with interaction terms
- Decision boundaries: what they are and how polynomial features create non-linear boundaries
- Regularization (Ridge/L2) in logistic regression to prevent overfitting
- Multiclass classification: One-vs-Rest (OvR) and Multinomial approaches
- Classification evaluation: Confusion Matrix, Sensitivity, Specificity, Precision, ROC curves, and AUC

Contents

1 Review: The Probabilistic View of Logistic Regression

1.1 Setting Up the Problem

In logistic regression, our response variable Y takes values 0 or 1 (binary outcomes). We model the **probability of success** $P(Y = 1|X)$ using a probabilistic framework.

Since Y is binary (0 or 1), each observation follows a **Bernoulli distribution**:

$$Y_i|X_i \sim \text{Bernoulli}(p_i)$$

where $p_i = P(Y_i = 1|X_i)$ is the probability of success for observation i.

Definition:

Bernoulli Distribution A random variable Y follows a Bernoulli distribution with parameter p if:

$$P(Y = y) = p^y(1 - p)^{1-y}$$

This gives:

- $P(Y = 1) = p$ (probability of success)
- $P(Y = 0) = 1 - p$ (probability of failure)

The binomial distribution with $n = 1$ is simply a Bernoulli distribution.

1.2 Linking Probability to Predictors

We link the probability p to our predictors through the **log-odds** (logit) function:

$$\log\left(\frac{p}{1-p}\right) = \beta_0 + \beta_1 X$$

Solving for p, we get the **sigmoid function**:

$$p = \frac{e^{\beta_0+\beta_1 X}}{1 + e^{\beta_0+\beta_1 X}}$$

This is the "exponentiated odds" form mentioned in the lecture.

1.3 The Likelihood Function

Given our data, we want to find the β values that make observing our data most likely. The likelihood function for a single observation is the Bernoulli PMF:

$$L(p|y) = p^y(1 - p)^{1-y}$$

Example:

Understanding the Bernoulli Likelihood Let's verify the likelihood makes intuitive sense:

- If $y = 1$ (success): $L = p^1(1-p)^0 = p$. We want p to be high.
- If $y = 0$ (failure): $L = p^0(1-p)^1 = 1-p$. We want p to be low (so $1-p$ is high).

The formula correctly captures what we want: high probability for what actually happened.

The total likelihood for all n observations is:

$$L(\boldsymbol{\beta}) = \prod_{i=1}^{n} p_i^{y_i}(1-p_i)^{1-y_i}$$

where the β parameters are hidden inside each p_i.

2 Maximum Likelihood Estimation (MLE)

2.1 The Estimation Process

To find the optimal β values, we:

1. Take the **logarithm** of the likelihood (easier to work with sums than products)
2. Take the **derivative** with respect to each β
3. Set the derivatives equal to **zero** and solve

Warning

No Closed-Form Solution!
Unlike linear regression where we can solve for $\hat{\beta} = (X^T X)^{-1} X^T Y$ directly, logistic regression has no closed-form solution. The equations are non-linear and must be solved **numerically**.

2.2 Numerical Optimization

Since we cannot solve analytically, we use numerical methods:

- **Gradient Descent**: Iteratively move in the direction that decreases the loss
- **Newton-Raphson Method**: Uses second derivatives for faster convergence

We typically minimize the **negative log-likelihood**, which is equivalent to maximizing the likelihood. This negative log-likelihood is called **Binary Cross-Entropy**:

$$\text{Loss} = -\sum_{i=1}^{n} [y_i \log(p_i) + (1 - y_i) \log(1 - p_i)]$$

Key Information

What Happens Under the Hood in sklearn
When you call `LogisticRegression().fit(X, y)`, sklearn is running an optimization algorithm (typically LBFGS or similar) to minimize the binary cross-entropy loss and find the optimal β coefficients.

3 Inference in Logistic Regression

Once we have estimates $\hat{\beta}$, we want to:

1. **Interpret** what these coefficients mean
2. **Quantify uncertainty** through confidence intervals
3. **Test hypotheses** about whether coefficients are significantly different from zero

3.1 Z-Statistics vs T-Statistics

3.2 Practical Implications

For confidence intervals:

- Linear regression: $\hat{\beta} \pm t_{\alpha/2,df} \times SE(\hat{\beta})$ (often ≈ 2)
- Logistic regression: $\hat{\beta} \pm \mathbf{1.96} \times SE(\hat{\beta})$ for 95% CI

The value 1.96 comes from the standard normal distribution (Z-distribution).

3.3 Using statsmodels for Inference

While `sklearn` is great for prediction, it does not provide standard errors directly. For statistical inference, use `statsmodels`:

```python
import statsmodels.formula.api as smf

# Fit logistic regression with formula interface
model = smf.logit("heart_disease ~ max_heart_rate", data=df).fit()

# View the summary with coefficients, standard errors, z-stats, p-values, and CIs
print(model.summary())
```

Listing 1: Logistic Regression with statsmodels

The output will include:

- **Coefficients** ($\hat{\beta}$)
- **Standard Errors** (from Fisher's Information)
- **Z-statistic**: $z = \hat{\beta}/SE(\hat{\beta})$
- **P-value**: for testing $H_0 : \beta = 0$
- **95% Confidence Interval**: $\hat{\beta} \pm 1.96 \times SE$

Key Information

Fisher's Information

The standard errors in logistic regression come from **Fisher's Information**, which measures the curvature of the log-likelihood function at its maximum. More curvature (sharper peak) means more certainty, hence smaller standard errors.

4 Interpreting Coefficients with Binary Predictors

When a predictor X is binary (0 or 1), interpretation becomes especially clear.

4.1 Setting Up: Reference Groups

Consider predicting heart disease based on biological sex:

- $X = 1$ if female
- $X = 0$ if male (the **reference group** or **baseline group**)

The model is:

$$\log\left(\frac{p}{1-p}\right) = \beta_0 + \beta_1 \cdot \text{Female}$$

4.2 Interpreting β_0 (Intercept)

When $X = 0$ (male):

$$\log(\text{Odds}_{\text{male}}) = \beta_0$$

So β_0 is the **log-odds of success for the reference group**.

Example:

Calculating β_0 from Data From the lecture, approximately 52% of males had heart disease:

- Probability: $\hat{p} = 0.52$
- Odds: $\frac{0.52}{1-0.52} = \frac{0.52}{0.48} \approx 1.08$
- Log-odds: $\ln(1.08) \approx 0.077$

So $\beta_0 \approx 0.077$ (or about 0.214 with exact numbers).

4.3 Interpreting β_1 (Slope for Binary Predictor)

β_1 represents the **difference in log-odds** comparing the indicated group (female) to the reference group (male).

When $X = 1$ (female):

$$\log(\text{Odds}_{\text{female}}) = \beta_0 + \beta_1$$

Therefore:

$$\beta_1 = \log(\text{Odds}_{\text{female}}) - \log(\text{Odds}_{\text{male}})$$

Example:

Calculating β_1 from Data From the lecture, approximately 25% of females had heart disease:

- Odds for females: $\frac{0.25}{0.75} = 0.33$
- Log-odds for females: $\ln(0.33) \approx -1.11$
- Difference: $-1.11 - 0.077 \approx -1.19$ (about -1.272 with exact numbers)

A negative β_1 means females have **lower** log-odds of heart disease than males.

4.4 Exponentiation: Odds Ratios

Raw coefficients are on the log-odds scale, which is hard to interpret. **Exponentiate** to get interpretable **odds ratios**:

- e^{β_0} = Odds for the reference group
- e^{β_1} = **Odds Ratio** (multiplicative change in odds)

Definition:

Odds Ratio Interpretation If $e^{\beta_1} = 0.28$:
"The odds of heart disease for females are **0.28 times** (or 28%) the odds for males."
Equivalently: "Females have 72% **lower odds** of heart disease compared to males."

Warning

Odds ≠ Probability!
Never say "72% lower probability." The relationship between odds ratios and probability changes is non-linear and depends on the baseline probability.
Correct: "The odds are 0.28 times as high" or "72% reduction in odds"
Incorrect: "72% reduction in probability"

5 Multiple Logistic Regression

Just like in linear regression, we can include multiple predictors:

$$\log\left(\frac{p}{1-p}\right) = \beta_0 + \beta_1 X_1 + \beta_2 X_2 + \cdots + \beta_p X_p$$

5.1 Key Points

1. **Interpretation**: Each β_j represents the change in log-odds for a one-unit increase in X_j, **holding all other predictors constant**.
2. **Same issues as linear regression**:
   - Multicollinearity (correlated predictors inflate variance)
   - Overfitting (too many predictors relative to sample size)
3. **Visualization**: Instead of fitting an S-curve to a scatter plot, we're fitting an "S-shaped hyperplane" in higher dimensions.

5.2 Interaction Terms

Interaction terms allow the effect of one predictor to **depend on** the value of another predictor.

Example:

Heart Disease: Age × Female Interaction Model:

$$\log(\text{Odds}) = \beta_0 + \beta_1 \cdot \text{Age} + \beta_2 \cdot \text{Female} + \beta_3 \cdot (\text{Age} \times \text{Female})$$

Let's separate this by sex:

For Males (Female = 0):

$$\log(\text{Odds})_{\text{male}} = \beta_0 + \beta_1 \cdot \text{Age}$$

- Intercept: β_0
- Slope for Age: β_1

For Females (Female = 1):

$$\log(\text{Odds})_{\text{female}} = \beta_0 + \beta_1 \cdot \text{Age} + \beta_2 + \beta_3 \cdot \text{Age}$$

$$= (\beta_0 + \beta_2) + (\beta_1 + \beta_3) \cdot \text{Age}$$

- Intercept: $\beta_0 + \beta_2$ (different from males!)
- Slope for Age: $\beta_1 + \beta_3$ (different from males!)

Coefficient Interpretations:

- β_1: Effect of age on log-odds **for males** (reference group)
- β_3: How much the age effect **differs** for females compared to males

If $\beta_3 = 0$, the age effect is the same for both sexes (no interaction). Testing $H_0 : \beta_3 = 0$ tests whether

the interaction is significant.

6 From Probabilities to Classifications

Logistic regression outputs **probabilities** $\hat{p} = P(Y = 1|X)$. To make actual **classifications** (predict 0 or 1), we need a **threshold**.

6.1 The 0.5 Threshold

The most common approach:

$$\hat{Y} = \begin{cases} 1 & \text{if } \hat{p} \geq 0.5 \\ 0 & \text{if } \hat{p} < 0.5 \end{cases}$$

Intuition: If the probability of success is at least 50%, predict success.

Key Information

What about exactly 0.5?
In practice, this rarely matters. You can round up, round down, or flip a coin. The exact handling of 0.5 typically has negligible impact on overall performance.

6.2 Extension to Multiple Classes (K > 2)

When you have 3+ classes, the 0.5 rule doesn't work—you're not guaranteed any class has probability > 0.5.

Solution: Plurality Wins

Predict the class with the **highest probability**, even if it's below 0.5.

Example with 3 classes:

- $P(\text{CS}) = 0.35$
- $P(\text{Stat}) = 0.40$
- $P(\text{Other}) = 0.25$

Prediction: **Stat** (highest probability)

7 Decision Boundaries

A **decision boundary** is the line (or curve) where the model switches from predicting one class to another.

7.1 Mathematical Derivation

The decision boundary occurs where $P(Y = 1) = 0.5$. Let's trace through the math:

1. $P(Y = 1) = 0.5$
2. Odds $= \frac{0.5}{1-0.5} = \frac{0.5}{0.5} = 1$
3. Log-odds $= \ln(1) = 0$

Therefore, the decision boundary is defined by:

$$\beta_0 + \beta_1 X_1 + \beta_2 X_2 + \cdots = 0$$

Definition:

Decision Boundary The decision boundary in logistic regression is the set of all points where:

$$X\boldsymbol{\beta} = 0$$

Equivalently, it's where the log-odds equals zero, the odds equals one, and the probability equals 0.5.

7.2 Linear vs Non-Linear Decision Boundaries

Linear Decision Boundary:

If the model only includes linear terms $(X_1, X_2, \ldots)$, the equation $\beta_0 + \beta_1 X_1 + \beta_2 X_2 = 0$ defines a **straight line** (or hyperplane in higher dimensions).

Non-Linear Decision Boundary:

If the model includes:

- **Polynomial terms**: $X_1^2, X_2^2, X_1^3, \ldots$
- **Interaction terms**: $X_1 \cdot X_2$

Then the decision boundary becomes a **curve** (or curved surface).

Example:

Creating a Circular Decision Boundary To create a circular decision boundary, include quadratic terms:

$$\log(\text{Odds}) = \beta_0 + \beta_1 X_1 + \beta_2 X_2 + \beta_3 X_1^2 + \beta_4 X_2^2 + \beta_5 X_1 X_2$$

Setting this equal to zero can give a circle, ellipse, or other conic section depending on the coefficient values.

Warning

Complexity vs Overfitting

More complex decision boundaries (from polynomial features) can capture more intricate patterns, but they also risk **overfitting**—fitting the training data too closely and performing poorly on new data.

8 Regularization in Logistic Regression

When we have many features or polynomial terms, the model can become overfit. **Regularization** prevents this by penalizing large coefficients.

8.1 The Regularized Loss Function

Standard logistic regression minimizes binary cross-entropy. With **Ridge (L2) regularization**, we add a penalty:

$$\text{Loss}_{\text{regularized}} = \underbrace{-\sum_{i=1}^{n}[y_i \log(p_i) + (1-y_i)\log(1-p_i)]}_{\text{Binary Cross-Entropy}} + \underbrace{\lambda \sum_{j=1}^{p} \beta_j^2}_{\text{L2 Penalty}}$$

- λ controls the **strength of regularization**
- The penalty is applied to all β_j **except** β_0 (the intercept)

8.2 Effect of λ

- $\lambda = 0$: No regularization (standard logistic regression)
- λ small: Weak regularization, coefficients can be large
- λ large: Strong regularization, coefficients shrink toward zero
- $\lambda \to \infty$: All coefficients become zero (except intercept)

8.3 sklearn's C Parameter

Important:

Understanding C in sklearn sklearn uses `C` instead of λ, where $C \approx 1/\lambda$:

- **Large C** (e.g., 100): **Weak** regularization (small λ). Model fits training data more closely. Risk of overfitting.
- **Small C** (e.g., 0.01): **Strong** regularization (large λ). Coefficients shrink. Simpler model. Risk of underfitting.

```
from sklearn.linear_model import LogisticRegression
from sklearn.model_selection import GridSearchCV

# Strong regularization (small C = large lambda)
model = LogisticRegression(penalty='l2', C=0.1)

# Find optimal C through cross-validation
param_grid = {'C': [0.001, 0.01, 0.1, 1, 10, 100]}
grid_search = GridSearchCV(
    LogisticRegression(penalty='l2'),
    param_grid,
    cv=5,
```

```
    scoring='accuracy'
)
grid_search.fit(X_train, y_train)
print(f"Best C: {grid_search.best_params_['C']}")
```

Listing 2: Regularized Logistic Regression in sklearn

8.4 Visualizing Regularization's Effect on Decision Boundaries

Imagine an overfit model with a very "wiggly" decision boundary that perfectly separates training points. Regularization smooths out this boundary, making it more generalizable:

- **Unregularized**: Complex, possibly overfit boundary
- **Regularized**: Smoother, more generalizable boundary

9 Multiclass Classification

When the response variable has **more than two categories** (e.g., CS major, Stats major, Other), we need to extend binary logistic regression.

We focus on **nominal** categories (no inherent order). For **ordinal** categories (e.g., low/medium/high), specialized ordinal regression methods exist.

9.1 Approach 1: Multinomial Logistic Regression

Choose one class as the **reference group** (e.g., "Other"). Build $K - 1$ separate log-odds models comparing each class to the reference.

For 3 classes (CS, Stat, Other):

$$\log\left(\frac{P(\text{CS})}{P(\text{Other})}\right) = \beta_0^{(1)} + \beta_1^{(1)} X_1 + \cdots$$

$$\log\left(\frac{P(\text{Stat})}{P(\text{Other})}\right) = \beta_0^{(2)} + \beta_1^{(2)} X_1 + \cdots$$

Note: These are fit simultaneously, not independently, ensuring probabilities sum to 1.

```
from sklearn.linear_model import LogisticRegression

# Multinomial is now the default in newer sklearn versions
model = LogisticRegression(multi_class='multinomial', solver='lbfgs')
model.fit(X_train, y_train)

# Predict probabilities for all classes
probs = model.predict_proba(X_test)  # Shape: (n_samples, n_classes)
```

Listing 3: Multinomial Logistic Regression in sklearn

9.2 Approach 2: One-vs-Rest (OvR)

Build K separate binary classifiers:

- Classifier 1: CS vs Not-CS
- Classifier 2: Stat vs Not-Stat
- Classifier 3: Other vs Not-Other

For prediction, run all classifiers and pick the one with highest probability/confidence.

```
from sklearn.multiclass import OneVsRestClassifier
from sklearn.linear_model import LogisticRegression

model = OneVsRestClassifier(LogisticRegression())
model.fit(X_train, y_train)
predictions = model.predict(X_test)
```

Listing 4: One-vs-Rest Classification

9.3 Softmax: Converting Scores to Probabilities

Both approaches produce "scores" for each class. To convert to proper probabilities (that sum to 1), we use the **Softmax function**:

$$P(Y = k|X) = \frac{e^{s_k}}{\sum_{j=1}^{K} e^{s_j}}$$

where s_k is the score (log-odds) for class k.

Key Information

Properties of Softmax

- All outputs are between 0 and 1
- All outputs sum to exactly 1
- Higher scores get higher probabilities
- The relative differences are preserved (monotonic transformation)

10 Classification Evaluation: The Confusion Matrix

Once we have a classifier, how do we evaluate its performance? Simple accuracy can be misleading, especially with imbalanced classes.

10.1 The Confusion Matrix

The confusion matrix organizes all possible prediction outcomes:

		Predicted	
		Negative (0)	Positive (1)
Actual	Negative (0)	**TN** (True Negative)	**FP** (False Positive)
	Positive (1)	**FN** (False Negative)	**TP** (True Positive)

Definition:

Confusion Matrix Terms

- **True Positive (TP)**: Actually positive, predicted positive. Correct!
- **True Negative (TN)**: Actually negative, predicted negative. Correct!
- **False Positive (FP)**: Actually negative, predicted positive. Type I Error.
- **False Negative (FN)**: Actually positive, predicted negative. Type II Error.

Example:

Medical Diagnosis Context Consider a test for cancer:

- **TP**: Patient has cancer, test says positive. (Detected the disease)
- **TN**: Patient is healthy, test says negative. (Correctly cleared)
- **FP**: Patient is healthy, test says positive. (False alarm, unnecessary anxiety)
- **FN**: Patient has cancer, test says negative. (Missed diagnosis—**dangerous!**)

```python
from sklearn.metrics import confusion_matrix, classification_report

y_pred = model.predict(X_test)

# Confusion matrix
cm = confusion_matrix(y_test, y_pred)
print("Confusion Matrix:")
print(cm)
# [[TN, FP],
#  [FN, TP]]

# Detailed report
print(classification_report(y_test, y_pred))
```

Listing 5: Computing Confusion Matrix in sklearn

11 Key Performance Metrics

Different metrics focus on different aspects of performance. The choice depends on the application and costs of different errors.

11.1 Sensitivity (Recall, True Positive Rate)

$$\text{Sensitivity} = \text{TPR} = \frac{TP}{TP + FN}$$

Question answered: Of all actually positive cases, what fraction did we correctly identify?

When it matters: When **missing a positive is costly**. In medical diagnosis, we don't want to miss cancer patients (minimize FN).

11.2 Specificity (True Negative Rate)

$$\text{Specificity} = \text{TNR} = \frac{TN}{TN + FP}$$

Question answered: Of all actually negative cases, what fraction did we correctly identify?

When it matters: When **false alarms are costly**. In spam filtering, we don't want to mark important emails as spam (minimize FP).

11.3 Precision (Positive Predictive Value)

$$\text{Precision} = \text{PPV} = \frac{TP}{TP + FP}$$

Question answered: Of all cases we predicted positive, what fraction are actually positive?

When it matters: When **acting on positive predictions is costly**. If treatment is expensive/harmful, we want predictions to be reliable.

11.4 False Positive Rate

$$\text{FPR} = 1 - \text{Specificity} = \frac{FP}{TN + FP}$$

Question answered: Of all actually negative cases, what fraction did we incorrectly classify as positive?

Warning

The Base Rate Problem (Bayes' Theorem)
Even with 99% sensitivity and 99% specificity, if a disease is very rare (e.g., 0.1% prevalence), most positive test results will be **false positives**!
This is why precision (PPV) can be low even with excellent sensitivity and specificity when the base rate is low.

12 The Threshold Trade-off

The classification threshold (default 0.5) is not sacred. Adjusting it creates a **trade-off** between sensitivity and specificity.

12.1 Lowering the Threshold (e.g., 0.5 → 0.3)

The model becomes more "eager" to predict positive:

- More True Positives (TP ↑) — **good**
- Fewer False Negatives (FN ↓) — **good**
- **Sensitivity increases**

But also:

- More False Positives (FP ↑) — **bad**
- Fewer True Negatives (TN ↓) — **bad**
- **Specificity decreases**

Use case: Medical screening where missing a case is unacceptable.

12.2 Raising the Threshold (e.g., 0.5 → 0.7)

The model becomes more "conservative" about predicting positive:

- Fewer False Positives (FP ↓) — **good**
- More True Negatives (TN ↑) — **good**
- **Specificity increases**

But also:

- Fewer True Positives (TP ↓) — **bad**
- More False Negatives (FN ↑) — **bad**
- **Sensitivity decreases**

Use case: When false positives are very costly (e.g., invasive surgery based on prediction).

Key Information

No Free Lunch

You cannot simultaneously maximize both sensitivity and specificity. Improving one typically hurts the other. The optimal threshold depends on the **relative costs** of false positives vs false negatives in your specific application.

13 ROC Curves and AUC

The **ROC Curve** (Receiver Operating Characteristic) visualizes the trade-off across **all possible thresholds**.

13.1 Constructing the ROC Curve

1. For each threshold from 0 to 1:
   - Calculate the True Positive Rate (TPR = Sensitivity)
   - Calculate the False Positive Rate (FPR = 1 - Specificity)
2. Plot each (FPR, TPR) pair as a point
3. Connect the points to form the curve

Axes:

- X-axis: False Positive Rate (FPR)
- Y-axis: True Positive Rate (TPR)

13.2 Interpreting the ROC Curve

- **Perfect classifier**: Goes from (0,0) to (0,1) to (1,1). It achieves 100% TPR with 0% FPR.
- **Random classifier**: The diagonal line from (0,0) to (1,1). TPR equals FPR at every threshold.
- **Good classifier**: Curves toward the upper-left corner (0,1).

13.3 AUC: Area Under the Curve

The **AUC** summarizes the entire ROC curve in a single number:

$$\text{AUC} = \int_0^1 \text{ROC}(x)\, dx$$

- **AUC = 1.0**: Perfect classifier
- **AUC = 0.5**: Random classifier (useless)
- **AUC = 0.8-0.9**: Good classifier
- **AUC < 0.5**: Worse than random (predictions are inverted)

Key Information

Probabilistic Interpretation of AUC
AUC equals the probability that a randomly chosen positive example is ranked higher (assigned higher probability) than a randomly chosen negative example.
AUC = 0.8 means: if you pick one positive and one negative case at random, there's an 80% chance the model assigns a higher probability to the positive case.

```
from sklearn.metrics import roc_curve, roc_auc_score
import matplotlib.pyplot as plt

```

```python
# Get predicted probabilities
y_proba = model.predict_proba(X_test)[:, 1]

# Calculate ROC curve
fpr, tpr, thresholds = roc_curve(y_test, y_proba)

# Calculate AUC
auc = roc_auc_score(y_test, y_proba)
print(f"AUC: {auc:.3f}")

# Plot
plt.figure(figsize=(8, 6))
plt.plot(fpr, tpr, label=f'ROC Curve (AUC = {auc:.3f})')
plt.plot([0, 1], [0, 1], 'k--', label='Random Classifier')
plt.xlabel('False Positive Rate')
plt.ylabel('True Positive Rate')
plt.title('ROC Curve')
plt.legend()
plt.grid(True, alpha=0.3)
plt.show()
```

Listing 6: ROC Curve and AUC in sklearn

14 Alternative Inference: Bootstrap and Permutation

Just as in linear regression, we have two approaches to inference:

14.1 Probabilistic Approach (Default)

Uses mathematical theory (Fisher's Information) to derive standard errors, confidence intervals, and p-values. This is what `statsmodels` provides.

Advantages:

- Fast computation
- Well-established theory

Disadvantages:

- Relies on asymptotic approximations (large sample)
- May not be accurate for small samples

14.2 Empirical Approach (Bootstrap/Permutation)

Bootstrap for Confidence Intervals:

1. Resample your data with replacement
2. Fit the model on each bootstrap sample
3. Calculate the coefficient
4. Repeat many times to get a distribution
5. Use percentiles for confidence intervals

Permutation for Hypothesis Testing:

1. Shuffle the Y values (break association with X)
2. Fit the model and record the coefficient
3. Repeat many times to get the null distribution
4. Compare observed coefficient to null distribution

Key Information

Advantages of Empirical Methods in Logistic Regression

In logistic regression, we don't have to worry about:

- Heteroscedasticity (variance depends on p, not separate)
- Normality of errors (no continuous errors)

The main assumption that matters is the correct specification of the model (right predictors, right functional form).

15 Practical Checklist

Before Your Midterm: Key Concepts Checklist

- ☐ Can you explain why logistic regression uses MLE instead of OLS?
- ☐ Do you understand why there's no closed-form solution (numerical methods needed)?
- ☐ Can you interpret β_0 for a model with a binary predictor (reference group concept)?
- ☐ Can you interpret β_1 as a difference in log-odds and e^{β_1} as an odds ratio?
- ☐ Do you understand the Z-distribution vs t-distribution distinction for inference?
- ☐ Can you write out how interaction terms create different slopes for different groups?
- ☐ Can you derive why $X\beta = 0$ is the decision boundary (from $P = 0.5$)?
- ☐ Do you understand how polynomial terms create non-linear decision boundaries?
- ☐ Can you explain regularization and the meaning of sklearn's C parameter?
- ☐ Do you know the difference between multinomial and one-vs-rest for multiclass?
- ☐ Can you calculate sensitivity, specificity, and precision from a confusion matrix?
- ☐ Do you understand the threshold trade-off (sensitivity vs specificity)?
- ☐ Can you interpret an ROC curve and explain what AUC measures?

Exam Tips from the Lecture

- For back-of-envelope calculations: use 2 for t-values and 1.96 for z-values
- Default answers if stuck: "cross-validation" or "it depends" or "MSE"
- Remember: logistic regression uses Z (not t) because variance is determined by p
- Coefficient interpretation: always talk about **odds**, not probability
- C in sklearn is **inverse** of λ: small C = strong regularization

16 Summary: Key Formulas

Logistic Regression Model

$$\log\left(\frac{p}{1-p}\right) = \beta_0 + \beta_1 X_1 + \cdots + \beta_p X_p$$

$$p = \frac{e^{X\beta}}{1+e^{X\beta}} = \frac{1}{1+e^{-X\beta}}$$

Loss Function

Binary Cross-Entropy:

$$\text{Loss} = -\sum_{i=1}^{n} [y_i \log(p_i) + (1-y_i)\log(1-p_i)]$$

With L2 Regularization:

$$\text{Loss}_{\text{reg}} = \text{Loss} + \lambda \sum_{j=1}^{p} \beta_j^2$$

Evaluation Metrics

$$\text{Sensitivity (TPR)} = \frac{TP}{TP+FN}$$

$$\text{Specificity (TNR)} = \frac{TN}{TN+FP}$$

$$\text{Precision (PPV)} = \frac{TP}{TP+FP}$$

$$\text{FPR} = \frac{FP}{TN+FP} = 1 - \text{Specificity}$$

$$\text{Accuracy} = \frac{TP+TN}{TP+TN+FP+FN}$$

Key Relationships

- Probability $\rightarrow$ Odds: Odds $= \frac{p}{1-p}$
- Odds $\rightarrow$ Probability: $p = \frac{\text{Odds}}{1+\text{Odds}}$
- Odds $\rightarrow$ Log-Odds: log(Odds)
- Coefficient $\rightarrow$ Odds Ratio: e^{β}
- Decision Boundary: where $X\beta = 0$ (equivalently $p = 0.5$)
- sklearn's C: $C \approx 1/\lambda$ (inverse regularization strength)