Logistic Regression Cheatsheet

BAN-501: Business Analytics

Contents

1 Odds and Log-Odds

1.1 From Probability to Odds

Given a probability $p = P(Y = 1)$, the **odds** are defined as:

$$\text{odds} = \frac{p}{1-p} = \frac{P(Y=1)}{P(Y=0)}$$

Interpretation: Odds represent the ratio of success to failure. An odds of 4 means "4 to 1 in favor."

1.1.1 Worked Examples

- $p = 0.2$: odds $= \frac{0.2}{1-0.2} = \frac{0.2}{0.8} = 0.25$ (4 to 1 against)
- $p = 0.5$: odds $= \frac{0.5}{1-0.5} = \frac{0.5}{0.5} = 1.0$ (even odds)
- $p = 0.8$: odds $= \frac{0.8}{1-0.8} = \frac{0.8}{0.2} = 4.0$ (4 to 1 in favor)

1.2 From Odds to Log-Odds (The Logit Function)

The **log-odds** (or **logit**) is the natural logarithm of the odds:

$$\text{logit}(p) = \log\left(\frac{p}{1-p}\right)$$

Key insight: The logit function maps probabilities from $[0, 1]$ to $(-\infty, +\infty)$.

1.3 Probability–Odds–Log-Odds Conversion Table

Probability (p)	**Odds**	**Log-Odds**
0.01	0.0101	-4.60
0.10	0.1111	-2.20
0.20	0.2500	-1.39
0.25	0.3333	-1.10
0.50	1.0000	0.00
0.75	3.0000	$+1.10$
0.80	4.0000	$+1.39$
0.90	9.0000	$+2.20$
0.99	99.0000	$+4.60$

Properties:

- Log-odds are **symmetric** around 0
- $p = 0.5$ corresponds to log-odds $= 0$
- Log-odds range from $-\infty$ to $+\infty$

2 The Logistic (Sigmoid) Function

The **logistic function** (also called the **sigmoid function**) is the inverse of the logit:

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

where z represents the log-odds (or any linear predictor).

2.1 Key Properties

1. **Bounded output**: Always produces values in $(0, 1)$—valid probabilities
2. **S-shaped**: Captures natural threshold behavior in classification
3. **Symmetric**: $\sigma(-z) = 1 - \sigma(z)$
4. **Smooth**: Differentiable everywhere (important for optimization)

2.2 Key Values

Input (z)	**Output** $\sigma(z)$
$-\infty$	$\to 0$
-2	0.119
-1	0.269
0	0.500
$+1$	0.731
$+2$	0.881
$+\infty$	$\to 1$

2.3 Relationship Between Logit and Sigmoid

The logit and sigmoid are **inverse functions**:

$$\sigma(\text{logit}(p)) = p \quad \text{and} \quad \text{logit}(\sigma(z)) = z$$

3 The Logistic Regression Model

3.1 Model Equation

Log-odds form:

$$\log\left(\frac{p}{1-p}\right) = \beta_0 + \beta_1 x_1 + \beta_2 x_2 + \cdots + \beta_k x_k$$

Probability form:

$$P(Y = 1 \mid \mathbf{x}) = \frac{1}{1 + e^{-(\beta_0 + \beta_1 x_1 + \cdots + \beta_k x_k)}}$$

For the simple case with one predictor:

$$P(Y = 1 \mid x) = \frac{1}{1 + e^{-(\beta_0 + \beta_1 x)}}$$

3.2 Coefficient Interpretation

- β_0 (intercept): The log-odds of $Y = 1$ when all predictors equal zero
- β_j (coefficient): The change in log-odds for a one-unit increase in x_j, holding other variables constant
- e^{β_j} (odds ratio): The multiplicative change in odds for a one-unit increase in x_j

Interpreting the odds ratio:

- $e^{\beta_j} > 1$: Odds increase (positive association)
- $e^{\beta_j} = 1$: No effect ($\beta_j = 0$)
- $e^{\beta_j} < 1$: Odds decrease (negative association)

4 Worked Prediction Example

Given: A fitted logistic regression model with:

- $\beta_0 = -2$ (intercept)
- $\beta_1 = 1.5$ (coefficient for x)
- New observation: $x = 2$

Step 1: Compute the linear predictor (log-odds)

$$z = \beta_0 + \beta_1 x = -2 + 1.5(2) = -2 + 3 = 1$$

Step 2: Apply the sigmoid function to get probability

$$P(Y = 1 \mid x = 2) = \sigma(1) = \frac{1}{1 + e^{-1}} = \frac{1}{1 + 0.368} = \frac{1}{1.368} \approx 0.731$$

Step 3: Interpret the result

- The predicted probability of $Y = 1$ is approximately 73.1%
- If using a threshold of 0.5, we would classify this observation as $\hat{Y} = 1$

Bonus: Interpret the coefficient

- $\beta_1 = 1.5$ means a one-unit increase in x increases log-odds by 1.5
- Odds ratio: $e^{1.5} \approx 4.48$, so a one-unit increase in x multiplies the odds by about 4.5

5 Python Code Snippets

5.1 Fitting Logistic Regression with Statsmodels

```python
import numpy as np
import statsmodels.api as sm

# Prepare data (add constant for intercept)
```

```python
X = sm.add_constant(X_train)

# Fit model
model = sm.Logit(
    endog=y_train,
    exog=X,
)
result = model.fit()

# View summary
print(result.summary())

# Extract coefficients
beta_0 = result.params[0]  # intercept
beta_1 = result.params[1]  # first coefficient

# Odds ratios
odds_ratios = np.exp(result.params)
```

5.2 Making Predictions

```python
# Prepare new data
X_new = sm.add_constant(X_test)

# Predict probabilities
probabilities = result.predict(X_new)

# Convert to class predictions (threshold = 0.5)
predictions = (probabilities >= 0.5).astype(int)
```

5.3 Manual Probability Calculation

```python
def sigmoid(z):
    """Compute the sigmoid/logistic function."""
    return 1 / (1 + np.exp(-z))

# Given coefficients and new x value
beta_0 = -2
beta_1 = 1.5
x_new = 2

# Compute probability
z = beta_0 + beta_1 * x_new  # linear predictor
prob = sigmoid(z)            # probability
print(f"P(Y=1 | x={x_new}) = {prob:.3f}")
```

6 Classification Metrics

6.1 Confusion Matrix

A confusion matrix summarizes classification results:

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

6.2 Key Metrics

Recall (Sensitivity, True Positive Rate):

$$\text{Recall} = \frac{TP}{TP + FN}$$

"Of all actual positives, what fraction did we correctly identify?"

Precision (Positive Predictive Value):

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

"Of all predicted positives, what fraction were actually positive?"

Specificity (True Negative Rate):

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

"Of all actual negatives, what fraction did we correctly identify?"

Accuracy:

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

F1 Score: Harmonic mean of precision and recall:

$$F_1 = 2 \cdot \frac{\text{Precision} \times \text{Recall}}{\text{Precision} + \text{Recall}} = \frac{2 \cdot TP}{2 \cdot TP + FP + FN}$$

6.3 Worked Example

Given confusion matrix:

	Pred. Neg.	**Pred. Pos.**
Actual Neg.	TN = 50	FP = 10
Actual Pos.	FN = 5	TP = 35

Calculations:

$$\text{Recall} = \frac{35}{35 + 5} = \frac{35}{40} = 0.875$$

$$\text{Precision} = \frac{35}{35 + 10} = \frac{35}{45} \approx 0.778$$

$$\text{Specificity} = \frac{50}{50 + 10} = \frac{50}{60} \approx 0.833$$

$$\text{Accuracy} = \frac{35 + 50}{35 + 50 + 10 + 5} = \frac{85}{100} = 0.85$$

$$F_1 = 2 \cdot \frac{0.778 \times 0.875}{0.778 + 0.875} = 2 \cdot \frac{0.681}{1.653} \approx 0.824$$

6.4 When to Prioritize Each Metric

Metric	Prioritize when...	Example
Recall	False negatives are costly	Disease screening, fraud detection
Precision	False positives are costly	Spam filtering, criminal conviction
F1 Score	Need balance between precision and recall	General classification tasks
Accuracy	Classes are balanced and errors are equally costly	Balanced datasets

6.5 Python Code for Metrics

```python
from sklearn.metrics import (
    confusion_matrix,
    classification_report,
    precision_score,
    recall_score,
    f1_score,
    accuracy_score,
)

# Compute confusion matrix
cm = confusion_matrix(
    y_true=y_test,
    y_pred=y_pred,
)
print("Confusion Matrix:")
print(cm)

# Individual metrics
print(f"Accuracy:  {accuracy_score(y_test, y_pred):.3f}")
print(f"Precision: {precision_score(y_test, y_pred):.3f}")
print(f"Recall:    {recall_score(y_test, y_pred):.3f}")
print(f"F1 Score:  {f1_score(y_test, y_pred):.3f}")

# Full classification report
print(classification_report(y_test, y_pred))
```

7 ROC Curve and AUC

7.1 Definitions

True Positive Rate (TPR) = Recall = Sensitivity:

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

False Positive Rate (FPR) = 1 − Specificity:

$$\text{FPR} = \frac{FP}{FP + TN}$$

7.2 ROC Curve

The **Receiver Operating Characteristic (ROC) curve** plots TPR vs. FPR at various classification thresholds.

How it's generated:

1. Start with predicted probabilities for all observations
2. Vary the classification threshold from 0 to 1
3. At each threshold, compute TPR and FPR
4. Plot (FPR, TPR) pairs as a curve

Key points on the ROC curve:

- $(0, 0)$: Threshold = 1 (predict all negative)
- $(1, 1)$: Threshold = 0 (predict all positive)
- $(0, 1)$: Perfect classifier
- Diagonal line: Random classifier (no skill)

7.3 AUC (Area Under the ROC Curve)

AUC measures the overall ability of the model to discriminate between classes.

Interpretation:

- AUC = probability that a randomly chosen positive instance is ranked higher than a randomly chosen negative instance
- AUC = 1.0: Perfect discrimination
- AUC = 0.5: No discrimination (random guessing)
- AUC $<$ 0.5: Worse than random (predictions are inverted)

AUC guidelines:

AUC Range	Interpretation
0.90–1.00	Excellent
0.80–0.90	Good
0.70–0.80	Fair
0.60–0.70	Poor
0.50–0.60	Fail

7.4 Python Code for ROC Curve

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

# Get predicted probabilities (not class predictions)
y_proba = model.predict_proba(X_test)[:, 1]

```

```python
# Compute ROC curve
fpr, tpr, thresholds = roc_curve(
    y_true=y_test,
    y_score=y_proba,
)

# Compute AUC
auc = roc_auc_score(
    y_true=y_test,
    y_score=y_proba,
)

# Plot ROC curve
fig, ax = plt.subplots(figsize=(6, 6))
ax.plot(fpr, tpr, label=f"ROC Curve (AUC = {auc:.3f})")
ax.plot([0, 1], [0, 1], "k--", label="Random Classifier")
ax.set_xlabel("False Positive Rate")
ax.set_ylabel("True Positive Rate")
ax.set_title("ROC Curve")
ax.legend()
plt.show()
```

8 Quick Reference

8.1 Key Formulas

Concept	Formula
Odds	$\frac{p}{1-p}$
Logit (log-odds)	$\log\left(\frac{p}{1-p}\right)$
Sigmoid	$\sigma(z) = \frac{1}{1+e^{-z}}$
Logistic regression	$P(Y=1) = \frac{1}{1+e^{-(\beta_0+\beta_1 x)}}$
Odds ratio	e^{β_j}
Recall	$\frac{TP}{TP+FN}$
Precision	$\frac{TP}{TP+FP}$
F1 Score	$\frac{2 \cdot TP}{2 \cdot TP + FP + FN}$
TPR (for ROC)	$\frac{TP}{TP+FN}$
FPR (for ROC)	$\frac{FP}{FP+TN}$

8.2 Coefficient Interpretation Checklist

1. Identify β_j from the model output
2. **Log-odds interpretation**: "A one-unit increase in x_j changes the log-odds by β_j"
3. Calculate odds ratio: e^{β_j}
4. **Odds ratio interpretation**: "A one-unit increase in x_j multiplies the odds by e^{β_j}"
5. If $e^{\beta_j} > 1$: odds increase; if $e^{\beta_j} < 1$: odds decrease