Lecture Notes on Performance Metrics in Logistic Regression

Your Name

February 28, 2025

Contents

1 Introduction **2**

2 Confusion Matrix and Performance Metrics **2**
2.1 Confusion Matrix 2
2.2 Precision, Recall, Accuracy, and F-Scores 2

3 Sensitivity vs. Specificity **3**
3.1 Definitions and Trade-offs 3
3.2 Real-World Example and Threshold Optimization 3

4 Effect of Threshold on Precision and Recall **4**

5 ROC Curve **4**
5.1 Derivation and Interpretation 4
5.2 Advantages and Drawbacks 4

6 Comparative Table of Metrics and Nomenclatures **4**

7 KS Statistic **5**
7.1 Derivation and Usage 5
7.2 Intuitive Understanding of the KS Statistic 5

8 Python Implementation Using the Titanic Dataset **5**

9 Conclusion **8**

1 Introduction

Logistic regression is one of the most widely used models for binary classification. In this lecture, we discuss a suite of performance metrics used to evaluate such models. These metrics include the confusion matrix, precision, recall (a.k.a. sensitivity or True Positive Rate), accuracy, and the family of F-scores. We also delve into sensitivity versus specificity, the ROC curve, and the Kolmogorov-Smirnov (KS) statistic. In the later section, we provide a comprehensive Python code example using a real-world dataset (the Titanic dataset) to illustrate these metrics in the context of logistic regression.

2 Confusion Matrix and Performance Metrics

2.1 Confusion Matrix

The confusion matrix is the foundation for many performance metrics. For a binary classifier, it is defined as:

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

From the confusion matrix, we can derive several key metrics.

2.2 Precision, Recall, Accuracy, and F-Scores

Accuracy:

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

Advantages: Simple and intuitive.
Disadvantages: Can be misleading in imbalanced datasets.

Precision:

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

Advantages: Indicates the proportion of positive predictions that are correct (useful when the cost of false positives is high).
Disadvantages: Does not account for false negatives.

Recall (Sensitivity, True Positive Rate):

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

Advantages: Measures the ability to capture all actual positives (important when missing a positive case is costly).
Disadvantages: May lead to many false positives if used in isolation.

F-Scores: The F-score is the harmonic mean of precision and recall. The most general form is the F_β score:

$$F_\beta = (1 + \beta^2) \cdot \frac{\text{Precision} \cdot \text{Recall}}{(\beta^2 \cdot \text{Precision}) + \text{Recall}}$$

Here, β determines the weight given to recall relative to precision.

- F_1 score: When $\beta = 1$, precision and recall are equally weighted.
- F_2 score: When $\beta = 2$, recall is weighted higher, which is useful when missing a positive case is far more severe than a false alarm.

Advantages: Balances precision and recall in a single metric.
Disadvantages: Does not capture true negatives and may hide the performance nuances of highly imbalanced data.

3 Sensitivity vs. Specificity

3.1 Definitions and Trade-offs

Sensitivity (Recall/TPR):

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

Specificity:

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

Sensitivity measures the proportion of true positives that are correctly identified, whereas specificity measures the proportion of true negatives correctly identified.

3.2 Real-World Example and Threshold Optimization

Consider a medical diagnostic test for a disease:

- **High sensitivity** is crucial if the cost of missing a positive case (i.e., a false negative) is very high.
- **High specificity** is desired if a false positive (i.e., misdiagnosing a healthy person) leads to unnecessary anxiety or costly follow-up tests.

In logistic regression, varying the decision threshold can adjust sensitivity and specificity. By plotting a sensitivity versus specificity curve, one can identify the optimal threshold—often determined by maximizing Youden's index:

$$J = \text{Sensitivity} + \text{Specificity} - 1$$

The threshold at which J is maximized is considered optimal.

4 Effect of Threshold on Precision and Recall

While sensitivity and specificity are useful, the choice of decision threshold in a logistic regression model directly affects both precision and recall. In general:

- **Lower thresholds** tend to classify more instances as positive. This generally increases recall (fewer false negatives) but may decrease precision (more false positives).
- **Higher thresholds** require stronger evidence before classifying an instance as positive. This typically increases precision (fewer false positives) but decreases recall (more false negatives).

A precision–recall vs. threshold plot is therefore useful to visualize this trade-off. By plotting both precision and recall as functions of the threshold, one can better understand how changing the threshold shifts the balance between these two important metrics. This information can help in selecting a threshold that best meets the requirements of a particular application.

5 ROC Curve

5.1 Derivation and Interpretation

The Receiver Operating Characteristic (ROC) curve plots the True Positive Rate (Sensitivity) against the False Positive Rate (FPR), where:

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

As the classification threshold varies, the TPR and FPR change, producing the ROC curve. The Area Under the ROC Curve (AUC) summarizes the overall ability of the model to discriminate between the classes.

5.2 Advantages and Drawbacks

- **Advantages:**
  - Threshold independent.
  - Provides a comprehensive view of trade-offs between TPR and FPR.
- **Drawbacks:**
  - Does not reflect the actual probability estimates.
  - May be overly optimistic in the presence of class imbalance.

6 Comparative Table of Metrics and Nomenclatures

The following table shows the relationships between different nomenclatures and their formulas:

Metric	Alternative Names	Formula
Accuracy	–	$\frac{TP+TN}{TP+TN+FP+FN}$
Precision	Positive Predictive Value (PPV)	$\frac{TP}{TP+FP}$
Recall	Sensitivity, True Positive Rate (TPR)	$\frac{TP}{TP+FN}$
Specificity	True Negative Rate (TNR)	$\frac{TN}{TN+FP}$
F_β Score	–	$(1+\beta^2) \cdot \frac{Precision \cdot Recall}{\beta^2 \cdot Precision + Recall}$

Table 1: Relationships between performance metrics.

7 KS Statistic

7.1 Derivation and Usage

The Kolmogorov-Smirnov (KS) statistic measures the maximum distance between the cumulative distribution functions (CDFs) of the scores for the positive and negative classes:

$$\text{KS} = \max_t |\text{CDF}_{\text{pos}}(t) - \text{CDF}_{\text{neg}}(t)|$$

It quantifies how well the model separates the two classes. A higher KS value indicates better separation.

7.2 Intuitive Understanding of the KS Statistic

Intuitively, the KS statistic represents the largest gap between the cumulative distributions of the positive and negative classes. Imagine plotting the cumulative percentage of positives and negatives as you lower the classification threshold. The maximum vertical gap between these curves indicates the threshold where the model best distinguishes between the two classes. In many applications, a KS value above 0.4 is considered good, indicating strong discrimination, though acceptable values depend on context.

8 Python Implementation Using the Titanic Dataset

Below is a complete Python code example that uses logistic regression to demonstrate the metrics discussed. The code:

- Loads and preprocesses the Titanic dataset.
- Trains a logistic regression model.
- Computes the confusion matrix, precision, recall, accuracy, and F-scores (including F_1 and F_2).
- Plots the ROC curve and computes AUC.
- Plots precision and recall as functions of the threshold.

- Plots the CDF curves for the positive and negative classes to visually illustrate the KS statistic.
- Computes the KS statistic.
- Compares the different metrics.

```
import numpy as np
import pandas as pd
import seaborn as sns
import matplotlib.pyplot as plt
from sklearn.linear_model import LogisticRegression
from sklearn.metrics import (confusion_matrix, accuracy_score,
    precision_score, recall_score,
                             f1_score, roc_curve, roc_auc_score,
    precision_recall_curve)
from sklearn.model_selection import train_test_split
from sklearn.preprocessing import LabelEncoder

# Load the Titanic dataset from seaborn
df = sns.load_dataset('titanic')
# Preprocessing: drop rows with missing values and encode categorical
    variables
df = df.dropna(subset=['age', 'fare', 'embarked', 'sex', 'survived'])
df['sex'] = LabelEncoder().fit_transform(df['sex'])
df['embarked'] = LabelEncoder().fit_transform(df['embarked'])

# Select features and target variable
features = ['pclass', 'age', 'fare', 'sex', 'embarked']
X = df[features]
y = df['survived']

# Split the dataset
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.3,
    random_state=42)

# Train logistic regression
model = LogisticRegression(max_iter=1000)
model.fit(X_train, y_train)
y_prob = model.predict_proba(X_test)[:, 1]
y_pred = (y_prob >= 0.5).astype(int)

# Compute confusion matrix and metrics
cm = confusion_matrix(y_test, y_pred)
acc = accuracy_score(y_test, y_pred)
prec = precision_score(y_test, y_pred)
rec = recall_score(y_test, y_pred)
f1 = f1_score(y_test, y_pred)

print("Confusion Matrix:\n", cm)
print("Accuracy:", acc)
print("Precision:", prec)
print("Recall (Sensitivity):", rec)
print("F1 Score:", f1)
```

```
44
45 # Compute F-beta scores
46 def f_beta(precision, recall, beta):
47     return (1 + beta**2) * (precision * recall) / ((beta**2 * precision) +
       recall)
48
49 f2 = f_beta(prec, rec, beta=2)
50 print("F2 Score:", f2)
51
52 # ROC Curve and AUC
53 fpr, tpr, thresholds = roc_curve(y_test, y_prob)
54 auc = roc_auc_score(y_test, y_prob)
55 plt.figure(figsize=(8,6))
56 plt.plot(fpr, tpr, label=f'ROC curve (area = {auc:.2f})')
57 plt.plot([0,1], [0,1], 'k--')
58 plt.xlabel('False Positive Rate (1 - Specificity)')
59 plt.ylabel('True Positive Rate (Sensitivity)')
60 plt.title('ROC Curve')
61 plt.legend()
62 plt.show()
63
64 # Precision-Recall vs. Threshold Plot
65 precisions, recalls, pr_thresholds = precision_recall_curve(y_test, y_prob
       )
66 plt.figure(figsize=(8,6))
67 # Note: precision_recall_curve returns arrays with len = len(thresholds)
       +1, so slice accordingly.
68 plt.plot(pr_thresholds, precisions[:-1], label='Precision')
69 plt.plot(pr_thresholds, recalls[:-1], label='Recall')
70 plt.xlabel('Threshold')
71 plt.ylabel('Value')
72 plt.title('Precision and Recall vs. Threshold')
73 plt.legend()
74 plt.show()
75
76 # Compute KS Statistic and Plot CDFs for Positive and Negative Classes
77 data = pd.DataFrame({'y_true': y_test, 'y_prob': y_prob})
78 data = data.sort_values(by='y_prob')
79 data['cum_positive'] = np.cumsum(data['y_true']) / data['y_true'].sum()
80 data['cum_negative'] = np.cumsum(1 - data['y_true']) / (1 - data['y_true'
       ]).sum()
81 ks_stat = np.max(np.abs(data['cum_positive'] - data['cum_negative']))
82 print("KS Statistic:", ks_stat)
83
84 plt.figure(figsize=(8,6))
85 plt.plot(data['y_prob'], data['cum_positive'], label='CDF Positive')
86 plt.plot(data['y_prob'], data['cum_negative'], label='CDF Negative')
87 max_idx = np.argmax(np.abs(data['cum_positive'] - data['cum_negative']))
88 plt.axvline(x=data['y_prob'].iloc[max_idx], color='red', linestyle='--',
       label='Max Difference Threshold')
89 plt.xlabel('Predicted Probability')
90 plt.ylabel('Cumulative Distribution')
91 plt.title('CDFs of Positive and Negative Classes')
92 plt.legend()
```

```python
plt.show()

# Comparative Analysis
metrics = {
    "Accuracy": acc,
    "Precision": prec,
    "Recall (Sensitivity)": rec,
    "F1 Score": f1,
    "F2 Score": f2,
    "AUC": auc,
    "KS Statistic": ks_stat,
    "Optimal Threshold": pr_thresholds[np.argmax(precisions[:-1] - recalls[:-1])]
}
print("\nComparative Metrics:")
for metric, value in metrics.items():
    print(f"{metric}: {value:.4f}")
```

Listing 1: Python Code for Logistic Regression Performance Analysis

9 Conclusion

In these notes we have:

- Derived and discussed key performance metrics based on the confusion matrix.
- Explored the trade-offs between sensitivity and specificity, and how to use their curve to determine an optimal threshold in logistic regression.
- Discussed the effect of varying the decision threshold on precision and recall, demonstrating that lower thresholds tend to increase recall (at the cost of precision) while higher thresholds improve precision (at the cost of recall).
- Reviewed the ROC curve as a threshold-independent measure of classifier performance.
- Provided a comparative table to clarify various nomenclatures.
- Discussed the KS statistic, both in derivation and through an intuitive lens, highlighting its use in identifying the best separation threshold between classes.
- Demonstrated all these concepts with a complete Python implementation on the Titanic dataset, including plots for the ROC curve, precision–recall vs. threshold, and the cumulative distribution functions of positive and negative classes.

These insights are critical for a comprehensive evaluation of logistic regression models in real-world applications.