Comprehensive Guide to Multiclass Classification Metrics

Understanding the pros and cons of the most popular classification metrics

Shreya Rao Follow

Published in Towards Data Science · 10 min read · Jun 2, 2022

Open in app Sign up Sign in

Introduction

In a previous article, I discussed the **binary classification metrics**. Within that article, I discussed the pros and cons of the most popular binary classification metrics. In this article, I will be discussing the pros and cons of the most popular multiclass classification metrics. We will be using the Iris dataset, which is a multiclass classification dataset, for illustrating the various metrics.

In the case of binary classification, it is conventional to call the positive class 1 and the negative class 0. However, in multiclass classification, there is no such convention. Instead, we need to specify which class is the positive class and which class is the negative class. For example, if we are classifying the Iris flower species, we can specify that Iris-virginica is the positive class and Iris-setosa and Iris-versicolor are the negative classes.

The rest of this article, will be in the context of Iris-virginica being the positive class and Iris-setosa and Iris-versicolor being the negative classes.

Understanding the Confusion matrix

The confusion matrix is a matrix that summarizes the performance of a machine learning model on a set of test data. It is often used to measure the performance of classification models, which aim to predict a categorical label for each input instance. The confusion matrix provides a detailed breakdown of the model's predictions, including the number of true positives, false positives, true negatives, and false negatives. These values can be used to calculate various performance metrics, such as accuracy, precision, recall, and the F1 score. The confusion matrix is a useful tool for understanding the strengths and weaknesses of a classification model and for identifying areas where the model may need to be improved.

The general form of the confusion matrix for a multiclass classification problem with n classes is an n x n matrix, where the rows represent the true class labels and the columns represent the predicted class labels. The diagonal elements of the matrix represent the number of correct predictions for each class, while the off-diagonal elements represent the number of incorrect predictions. For example, in a three-class classification problem, the confusion matrix might look like this:

	Class 1	Class 2	Class 3
Class 1	True positives for class 1	False positives for class 1	False positives for class 1
Class 2	False positives for class 2	True positives for class 2	False positives for class 2
Class 3	False positives for class 3	False positives for class 3	True positives for class 3

In this table, the rows represent the true class labels, the columns represent the predicted class labels, and the diagonal elements represent the number of correct predictions for each class. The off-diagonal elements represent the number of incorrect predictions, with the false positives for each class being the number of instances that were incorrectly classified as belonging to that class.

For the Iris dataset, the confusion matrix would be a 3 x 3 matrix, with the rows representing the true class labels (Iris-setosa, Iris-versicolor, and Iris-virginica) and the columns representing the predicted class labels. The diagonal elements would represent the number of correct predictions for each class, while the off-diagonal elements would represent the number of incorrect predictions. For example, if the model correctly classified 50 instances of Iris-setosa, 40 instances of Iris-versicolor, and 30 instances of Iris-virginica, but incorrectly classified 10 instances of Iris-setosa as Iris-versicolor and 5 instances of Iris-versicolor as Iris-virginica, the confusion matrix would look like this:

	Iris-setosa	Iris-versicolor	Iris-virginica
Iris-setosa	50	10	0
Iris-versicolor	0	40	5
Iris-virginica	0	0	30

In this table, the diagonal elements represent the number of correct predictions for each class, while the off-diagonal elements represent the number of incorrect predictions. The false positives for each class can be calculated by summing the off-diagonal elements in the column for that class. For example, the false positives for Iris-setosa would be 0, the false positives for Iris-versicolor would be 10, and the false positives for Iris-virginica would be 5.

Accuracy

Accuracy is a commonly used metric in machine learning that measures the proportion of correct predictions made by a model. It is calculated by dividing the number of correct predictions by the total number of predictions. In the context of a multiclass classification problem, accuracy can be calculated by summing the diagonal elements of the confusion matrix and dividing by the total number of instances. For example, in the case of the Iris dataset, the accuracy would be calculated as:

$$\text{Accuracy} = \frac{50 + 40 + 30}{50 + 10 + 0 + 0 + 40 + 5 + 0 + 0 + 30} = 0.86$$

In this example, the accuracy would be 0.86, or 86%, which means that the model correctly classified 86% of the instances in the Iris dataset. Accuracy is a useful metric for evaluating the performance of a machine learning model, but it may not be the most appropriate metric in cases where the classes are imbalanced or where the cost of false positives and false negatives is different.

Precision and recall for multiclass classification

Precision and recall are two metrics that are commonly used in machine learning to evaluate the performance of a classification model. Precision is the proportion of true positives among the instances that the model predicted as positive, while recall is the proportion of true positives among the instances that actually belong to the positive class. The formulas for calculating precision and recall are:

$$\text{Precision} = \frac{\text{True Positives}}{\text{True Positives} + \text{False Positives}}$$

$$\text{Recall} = \frac{\text{True Positives}}{\text{True Positives} + \text{False Negatives}}$$

In the context of a multiclass classification problem, we can calculate precision and recall for each class by treating the class as the positive class and the other classes as the negative class. For example, in the case of the Iris dataset, we can calculate the precision and recall for the Iris-virginica class as follows:

$$\text{Precision}_{\text{Iris-virginica}} = \frac{30}{0 + 5 + 0 + 30} = 0.86$$

$$\text{Recall}_{\text{Iris-virginica}} = \frac{30}{0 + 0 + 30} = 1.0$$

In this example, the precision for the Iris-virginica class is 0.86, or 86%, which means that the model correctly predicted 86% of the instances that belong to the Iris-virginica class. The recall for the Iris-virginica class is 1.0, or 100%, which means that the model correctly identified all of the instances that belong to the Iris-virginica class.

Micro-average precision and recall

Micro-average precision and recall are a variant of precision and recall that aggregate the true positives, false positives, and false negatives across all classes to calculate the overall precision and recall. The formulas for calculating micro-average precision and recall are:

$$\text{Micro-average Precision} = \frac{\sum \text{True Positives}}{\sum \text{True Positives} + \sum \text{False Positives}}$$

$$\text{Micro-average Recall} = \frac{\sum \text{True Positives}}{\sum \text{True Positives} + \sum \text{False Negatives}}$$

In the context of a multiclass classification problem, we can calculate micro-average precision and recall by summing the true positives, false positives, and false negatives for all classes and then dividing the total number of true positives by the total number of true positives plus the total number of false positives or false negatives. For example, in the case of the Iris dataset, we can calculate the micro-average precision and recall as follows:

$$\text{Micro-average Precision} = \frac{50 + 40 + 30}{50 + 40 + 30 + 10 + 0 + 0 + 5 + 0} = 0.86$$

$$\text{Micro-average Recall} = \frac{50 + 40 + 30}{50 + 40 + 30 + 0 + 0 + 0 + 0 + 0} = 1.0$$

In this example, the micro-average precision is 0.86, or 86%, which means that the model correctly predicted 86% of the instances that belong to the positive class. The micro-average recall is 1.0, or 100%, which means that the model correctly identified all of the instances that belong to the positive class.

Macro-average precision and recall

Macro-average precision and recall are another variant of precision and recall that aggregate the true positives, false positives, and false negatives across all classes to calculate the overall precision and recall. The formulas for calculating macro-average precision and recall are:

$$\text{Macro-average Precision} = \frac{\sum \text{Precision}_{\text{class}}}{\text{Number of classes}}$$

$$\text{Macro-average Recall} = \frac{\sum \text{Recall}_{\text{class}}}{\text{Number of classes}}$$

In the context of a multiclass classification problem, we can calculate macro-average precision and recall by summing the precision and recall for each class and then dividing by the number of classes. For example, in the case of the Iris dataset, we can calculate the macro-average precision and recall as follows:

$$\text{Macro-average Precision} = \frac{0.86}{3} = 0.29$$

$$\text{Macro-average Recall} = \frac{1.0}{3} = 0.33$$

In this example, the macro-average precision is 0.29, or 29%, which means that the model correctly predicted 29% of the instances that belong to the positive class. The macro-average recall is 0.33, or 33%, which means that the model correctly identified 33% of the instances that belong to the positive class.

F1 score

The F1 score is a metric that combines precision and recall into a single score. It is calculated by taking the harmonic mean of precision and recall, which is a weighted average that gives more weight to the lower value. The formula for calculating the F1 score is:

$$\text{F1 score} = \frac{2 \times \text{Precision} \times \text{Recall}}{\text{Precision} + \text{Recall}}$$

In the context of a multiclass classification problem, we can calculate the F1 score for each class by treating the class as the positive class and the other classes as the negative class. For example, in the case of the Iris dataset, we can calculate the F1 score for the Iris-virginica class as follows:

$$\text{F1 score}_{\text{Iris-virginica}} = \frac{2 \times 0.86 \times 1.0}{0.86 + 1.0} = 0.92$$

In this example, the F1 score for the Iris-virginica class is 0.92, or 92%, which means that the model correctly predicted 92% of the instances that belong to the Iris-virginica class.

Micro-average F1 score

Micro-average F1 score is a variant of the F1 score that aggregates the true positives, false positives, and false negatives across all classes to calculate the overall F1 score. The formula for calculating micro-average F1 score is:

$$\text{Micro-average F1 score} = \frac{2 \times \text{Micro-average Precision} \times \text{Micro-average Recall}}{\text{Micro-average Precision} + \text{Micro-average Recall}}$$

In the context of a multiclass classification problem, we can calculate micro-average F1 score by summing the true positives, false positives, and false negatives for all classes and then dividing the total number of true positives by the total number of true positives plus the total number of false positives or false negatives. For example, in the case of the Iris dataset, we can calculate the micro-average F1 score as follows:

$$\text{Micro-average F1 score} = \frac{2 \times 0.86 \times 1.0}{0.86 + 1.0} = 0.92$$

In this example, the micro-average F1 score is 0.92, or 92%, which means that the model correctly predicted 92% of the instances that belong to the positive class.

Macro-average F1 score

Macro-average F1 score is another variant of the F1 score that aggregates the true positives, false positives, and false negatives across all classes to calculate the overall F1 score. The formula for calculating macro-average F1 score is:

$$\text{Macro-average F1 score} = \frac{\sum \text{F1 score}_{\text{class}}}{\text{Number of classes}}$$

In the context of a multiclass classification problem, we can calculate macro-average F1 score by summing the F1 score for each class and then dividing by the number of classes. For example, in the case of the Iris dataset, we can calculate the macro-average F1 score as follows:

$$\text{Macro-average F1 score} = \frac{0.92}{3} = 0.31$$

In this example, the macro-average F1 score is 0.31, or 31%, which means that the model correctly predicted 31% of the instances that belong to the positive class.

Cohen's kappa score for multiclass classification

Cohen's kappa coefficient is a statistic that is used to measure the agreement between two raters (or classifiers) who are classifying items into mutually exclusive categories. The formula for calculating Cohen's kappa score is:

$$\kappa = \frac{p_o - p_e}{1 - p_e}$$

In a multiclass classification problem, we can calculate Cohen's kappa score by treating the ground truth labels as one rater and the predicted labels as the other rater. For example, in the case of the Iris dataset, we can calculate Cohen's kappa score as follows:

$$\kappa = \frac{0.86 - 0.33}{1 - 0.33} = 0.79$$

In this example, the Cohen's kappa score is 0.79, which indicates a high level of agreement between the ground truth labels and the predicted labels.

Hamming loss for multiclass classification

Hamming loss is a metric that measures the fraction of incorrect predictions made by a classifier. The formula for calculating Hamming loss is:

$$\text{Hamming loss} = \frac{\text{Number of incorrect predictions}}{\text{Total number of predictions}}$$

In a multiclass classification problem, we can calculate Hamming loss by summing the number of incorrect predictions for all classes and dividing by the total number of predictions. For example, in the case of the Iris dataset, we can calculate Hamming loss as follows:

$$\text{Hamming loss} = \frac{10 + 0 + 5 + 0}{150} = 0.10$$

In this example, the Hamming loss is 0.10, which means that the model incorrectly predicted 10% of the instances in the Iris dataset.

Matthews correlation coefficient for multiclass classification

Matthews correlation coefficient (MCC) is a metric that measures the quality of a binary classification model. The formula for calculating MCC is:

$$\text{MCC} = \frac{TP \times TN - FP \times FN}{\sqrt{(TP + FP)(TP + FN)(TN + FP)(TN + FN)}}$$

In a multiclass classification problem, we can calculate MCC by treating each class as a binary classification problem and averaging the MCC for each class. For example, in the case of the Iris dataset, we can calculate MCC as follows:

$$\text{MCC}_{\text{Iris-setosa}} = \frac{50 \times 90 - 10 \times 0}{\sqrt{(50 + 10)(50 + 0)(90 + 10)(90 + 0)}} = 0.83$$

$$\text{MCC}_{\text{Iris-versicolor}} = \frac{40 \times 85 - 10 \times 5}{\sqrt{(40 + 10)(40 + 5)(85 + 10)(85 + 5)}} = 0.76$$

$$\text{MCC}_{\text{Iris-virginica}} = \frac{30 \times 95 - 5 \times 0}{\sqrt{(30 + 5)(30 + 0)(95 + 5)(95 + 0)}} = 0.86$$

$$\text{Average MCC} = \frac{0.83 + 0.76 + 0.86}{3} = 0.82$$

In this example, the average MCC is 0.82, which indicates a high level of agreement between the ground truth labels and the predicted labels.

Log loss for multiclass classification

Log loss (also known as logarithmic loss or cross-entropy loss) is a metric that measures the performance of a classification model by calculating the negative log-likelihood of the true labels given the predicted probabilities. The formula for calculating log loss is:

$$\text{Log loss} = -\frac{1}{N} \sum_{i=1}^{N} \sum_{j=1}^{M} y_{ij} \log(p_{ij})$$

In a multiclass classification problem, we can calculate log loss by summing the log loss for each class and dividing by the number of classes. For example, in the case of the Iris dataset, we can calculate log loss as follows:

$$\text{Log loss}_{\text{Iris-setosa}} = -\frac{1}{150} \sum_{i=1}^{150} \sum_{j=1}^{3} y_{ij} \log(p_{ij})$$

$$\text{Log loss}_{\text{Iris-versicolor}} = -\frac{1}{150} \sum_{i=1}^{150} \sum_{j=1}^{3} y_{ij} \log(p_{ij})$$

$$\text{Log loss}_{\text{Iris-virginica}} = -\frac{1}{150} \sum_{i=1}^{150} \sum_{j=1}^{3} y_{ij} \log(p_{ij})$$

$$\text{Average log loss} = \frac{0.12 + 0.15 + 0.10}{3} = 0.12$$

In this example, the average log loss is 0.12, which indicates a high level of agreement between the ground truth labels and the predicted probabilities.

Conclusion

In conclusion, there are many metrics that can be used to evaluate the performance of a multiclass classification model, including the confusion matrix, accuracy, precision and recall, micro-average and macro-average precision and recall, F1 score, micro-average and macro-average F1 score, Cohen's kappa score, Hamming loss, Matthews correlation coefficient, and log loss. Each of these metrics has its pros and cons, and it is important to choose the appropriate metric for the task at hand and to interpret the results in the context of the problem being solved.

Shreya Rao
Machine Learning Engineer | Data Scientist | Ex-Amazon | Ex-Walmart | Ex-Flipkart | Ex-Ola | Ex-Mu Sigma | Ex-Fractal | Ex-Quantiphi | Ex-Infosys | Ex-Wipro | Ex-Hexaware | Ex-Cognizant
Follow

Machine Learning Data Science Statistics Classification Metrics
Sign up for The Weekly Struggle
Your weekly digest of the most challenging data science problems, projects, and resources. Delivered to your inbox every Monday.
Sign up
Sign in with Google
Sign in with Facebook
Sign in with Twitter
Sign in with LinkedIn
Sign in with Apple
Sign in with email
Sign up with email
Sign up with Google
Sign up with Facebook
Sign up with Twitter
Sign up with LinkedIn
Sign up with Apple
Sign up with email
Sign up for the Medium Newsletter
Sign up for the Medium Newsletter
Sign up for the Medium Newsletter