More about evaluation metrics for classification models

You have learned much about classification models, from logistic regression to Naive Bayes, and you will encounter yet others elsewhere in this course. Classification tasks are among the most common applications of machine learning. Knowing these techniques will empower you to take on data challenges from fraud detection to predicting stock market events and World Cup winners.

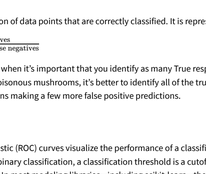
In this reading, you will reexamine some different ways to evaluate the performance of classification models, and also learn some new ones. You will review the confusion matrix and how it relates to accuracy, precision, and recall. (To revisit this information, please review Key metrics to assess logistic regression results and Common logistic regression metrics in Python.) You'll also learn about model evaluation using receiver operating characteristic (ROC) curves and area under the ROC curve (AUC), as well as F_1 score and F_β score.

Evaluation metrics for classification models

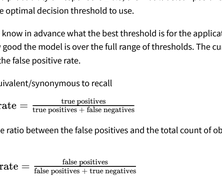
Previously you learned about linear regression models, their assumptions, theory, and use cases. You also learned how to evaluate these models using metrics like R^2, mean squared error (MSE), root mean squared error (RMSE), and mean absolute error (MAE). These metrics are all useful when evaluating the error of a prediction on a continuous variable.

You also learned about logistic regression, which does not predict on a continuous variable, but rather on a binary variable. It predicts a class. As such, it is a type of classification model.

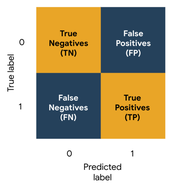
Classification models cannot be evaluated using the same metrics as linear regression models. Consider why. With a linear regression model, your model predicts a continuous value that has a unit label (e.g., dollars, kilograms, minutes, etc.) and your evaluation pertains to the residuals of the model's predictions—the difference between predicted and actual values:



But with a binary logistic regression, for example, your observations are represented by two classes; they are either one value or another. The model predicts a probability, and assigns observations to a class based on that probability.



Your evaluation must therefore pertain to the class assignment of your model. There are a number of such evaluation metrics that can be used with classification models. Some of these were introduced previously: accuracy, precision, and recall. These can all be derived from a confusion matrix, which is a graphical representation of your model.



Accuracy

Accuracy is the proportion of data points that are correctly classified. It is an overall representation of model performance, represented as:

$$Accuracy = \frac{\text{true positives + true negatives}}{\text{total predictions}}$$

Accuracy is often unsuitable to use when there is a class imbalance in the data, because it's possible for a model to have high accuracy by predicting the majority class every time. In such a case, the model would score well, but it may not be a useful model.

Precision

Precision measures the proportion of positive predictions that are true positives. It is represented as:

$$Precision = \frac{\text{true positives}}{\text{true positives + false positives}}$$

Precision is a good metric to use when it's important to avoid false positives. For example, if your model is designed to initially screen out ineligible loan applicants before a human review, then it's best to err on the side of caution and not automatically disqualify people before a person can review the case more carefully.

Recall

Recall measures the proportion of data points that are correctly classified. It is represented as:

$$Recall = \frac{\text{true positives}}{\text{true positives + false negatives}}$$

Recall is a good metric to use when it's important that you identify as many True responders as possible. For example, if your model is identifying poisonous mushrooms, it's better to identify all of the true occurrences of poisonous mushrooms, even if that means making a few more false positive predictions.

ROC curves

Receiver operating characteristic (ROC) curves visualize the performance of a classifier at different classification thresholds. In the context of binary classification, a classification threshold is a cutoff for differentiating the positive class from the negative class. In most modeling libraries—including scikit-learn—the default probability threshold is 0.5 (i.e., if a sample's predicted probability of response is ≥ 0.5, then it's labeled "positive"), but there are some cases where 0.5 might not be the optimal decision threshold to use.

Because you don't always know in advance what the best threshold is for the application, it may make sense to use an ROC curve to capture how good the model is over the full range of thresholds. The curve is represented by a plot of the true positive rate against the false positive rate.

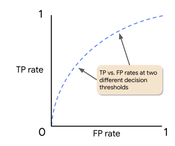
1. **True Positive Rate:** Equivalent/synonymous to recall

$$\text{True positive rate} = \frac{\text{true positives}}{\text{true positives + false negatives}}$$

2. **False Positive Rate:** The ratio between the false positives and the total count of observations that should be predicted as False

$$\text{False positive rate} = \frac{\text{false positives}}{\text{false positives + true negatives}}$$

For each point on an ROC curve, the horizontal and vertical coordinates represent the false positive rate and the true positive rate at the corresponding threshold.

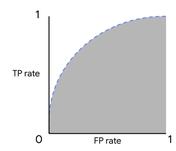


The false positive rate and true positive rate change together over the different thresholds.

How does the curve appear like for an ideal model? An ideal model perfectly separates all negatives from all positives, and gives all real positive cases a very high probability and all real negative cases a very low probability. So, imagine starting from a threshold just above zero: 0.001. In this case, it's likely that all real positives would be captured and there would be very few—if any—false negatives, because for a model to label a sample "negative," its predicted probability must be < 0.001. The true positive rate would be ≈ 1, and false positive rate ≈ 0 (refer to the formulas above). Graphically, the more that the ROC curve hugs the top left corner of the plot, the better the model is at classifying the data.

AUC

AUC is a measure of the two-dimensional area underneath an ROC curve. AUC provides an aggregate measure of performance across all possible classification thresholds. One way to interpret AUC is to consider it as the probability that the model ranks a random positive sample more highly than a random negative sample. AUC ranges in value from 0.0 to 1.0. In the following example, the AUC is the shaded region below the dotted curve.



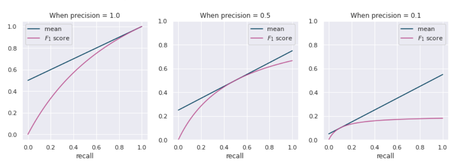
F_1 score

F_1 score is a measurement that combines both precision and recall into a single expression, giving each equal importance. It is calculated as:

$$F_1 = 2 \cdot \frac{\text{precision} \cdot \text{recall}}{\text{precision} + \text{recall}}$$

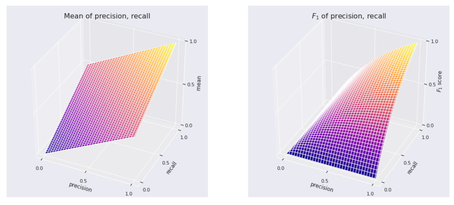
This combination is known as the harmonic mean. F_1 score can range [0, 1], with zero being the worst and one being the best. The idea behind this metric is that it penalizes low values of either metric, which prevents one very strong factor—precision or recall—from "carrying" the other, when it is weaker.

The following figure illustrates a comparison of the F_1 score to the mean of precision and recall. In each case, precision is held constant while recall ranges from 0 to 1.



The F_1 score never exceeds the mean. In fact, it is only equal to the mean in a single case: when precision equals recall. The more one score diverges from the other, the more F_1 score penalizes. (Note that you could swap precision and recall values in this experiment and the scores would be the same.)

Plotting the means and F_1 scores for all values of precision against all values of recall results in two planes.



While the coordinate plane of the mean is flat, the plane of the F_1 score is pulled further downward the more one score diverges from the other. This penalizing effect makes F_1 score a useful measurement of model performance.

F_β score

What if you still want to capture both precision and recall in a single metric, but you consider one more important than the other? There's a metric for that! It's called F_β score (pronounced F-beta). In an F_β score, β is a factor that represents how many times more important recall is compared to precision. In the case of F_1 score, $\beta = 1$, and recall is therefore 1x as important as precision (i.e., they are equally important). However, an F_2 score has $\beta = 2$, which means recall is twice as important as precision; and if precision is twice as important as recall, then $\beta = 0.5$. The formula for F_β score is:

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

You can assign whatever value you want to β. However, in scikit-learn, while most modules have built-in F_1 and F_β scorers, some modules may only have an F_1 scorer and require you to define your own scoring function if you want to set a custom β value.

Key takeaways

- There are many metrics used to evaluate binary classification models, including accuracy, precision, recall, ROC curve, AUC, and F score.
- Accuracy captures overall model performance, while precision measures true positives and recall measures false negatives.
- F_β score combines precision and recall into a single metric.
- You can set β to any value you wish. β is a factor that determines how many times more important recall is than precision in the score.
- F_1 score is simply an F_β score where $\beta = 1$. For F_1 score, precision is equally important to recall.

Resources for more information

- Evaluation metrics for classification models in scikit-learn: scikit-learn documentation for evaluation metrics used for classification

Mark as completed

Like **Dislike** **Report an issue**