Evaluating Model Quality

Evaluation metrics give us a way to quantitatively evaluate how well our model is performing. The model's performance on the training data is evaluated to get the training set accuracy, while its performance on the test data is evaluated to get the test data accuracy when the model predicts the targets of previously unseen examples. Evaluation on test data helps us to know the true performance measure of our model.

The learning problem determines the type of evaluation metric to use. As an example, for regression prediction problems, it is common to use the root mean squared error (RMSE) to evaluate the magnitude of the error made by the model. For classification problems, one of the common evaluation metrics is to use a confusion matrix to get a picture of how many samples are correctly classified or misclassified. From the confusion matrix, it is possible to derive other useful metrics for evaluating classification problems such as accuracy, precision, and recall.

The following are the evaluation metrics for machine learning that we will consider in this text:

Classification

- Confusion matrix
- Area under ROC curve (AUC-ROC)

Regression

- Root mean squared error (RMSE)
- R-squared (R^2)

Let's go through each.

Classification Evaluation Metrics

In this section, we'll briefly explain performance metrics for classification machine learning tasks.

Confusion Matrix

The confusion matrix is a popular evaluation metric for gleaning insights into the performance of a classification supervised machine learning model. It is represented as a table with grid-like cells. In the case of a two-class classification problem, the columns

of the grid are the actual positive and negative class values of the target feature, while the rows are the predicted positive and negative class values of the targets. This is illustrated in Figure 14-7.

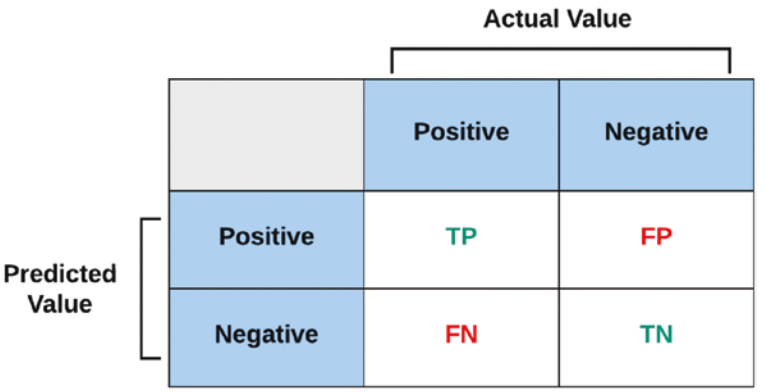

Figure 14-7. *Confusion matrix*

There are four primary values that can be gotten directly from examining the confusion matrix, and they are the ***true positive***, the ***false positive***, the ***true negative***, and the ***false negative*** *values*. Let's examine each of them briefly:

- True positive: True positive is the number of samples predicted to be positive (or true) when the actual class is positive.
- False positive: False positive is the number of samples predicted as positive (or true) when the actual class is negative.
- True negative: True negative is the number of samples predicted to be negative (or false) and the actual class is negative.
- False negative: False negative is the number of samples predicted to be negative (or false) when the actual class is positive.

From the four primary values, we have three other measures that provide more information on the performance of our model. These are accuracy, the positive predictive value (or precision), and sensitivity (or recall). Let's explain them briefly:

- Accuracy: Accuracy is the fraction of correct predictions made by the learning algorithm. It is represented as the ratio of the sum of true positive, *TP*, and true negative, *TN*, to the total population.

$$accuracy = \frac{TP + TN}{TP + FP + FN + TN}$$

- Precision or positive predictive value: Precision is the ratio of true positive, *TP*, to the sum of true positive, *TP*, and false positive, *FP*. In other words, precision measures the fraction of results that are correctly predicted as positive over all the results that the algorithm predicts as positive. The sum *TP* + *FP* is also called the predicted positive condition.

$$precision = \frac{TP}{TP + FP}$$

- Recall or sensitivity: Recall is the ratio of true positive, *TP*, to the sum of true positive, *TP*, and false negative, *FN*. In other words, recall retrieves the fraction of results that are correctly predicted as positive over all the results that are positive. The sum *TP* + *FN* is also known as condition positive.

$$recall = \frac{TP}{TP + FN}$$

To put this concept together, let's revisit the example heart disease dataset. Suppose we are to predict if a patient will be diagnosed with a heart disease or not, assume we have 50 samples in the dataset, of which 20 are diagnosed with heart disease and the remaining 30 are not. Of the 30 samples that do not have a disease diagnosis, the learning algorithm rightly identifies 25, while of the 20 samples that have a disease diagnosis, the learning algorithm correctly identifies 15.

Let's represent this information in a confusion matrix (see Figure 14-8) and calculate the necessary statistical measures to evaluate the algorithm performance.

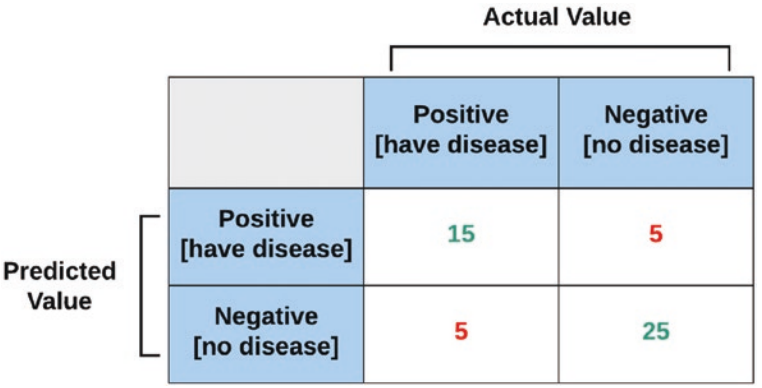

Figure 14-8. *Confusion matrix example*

From the data in Figure 14-8, we can calculate the ***accuracy***, ***precision***, and ***recall***.

- Accuracy =

$$\frac{15+25}{15+5+5+25}=\frac{40}{50}=\frac{4}{5}$$

- Precision =

$$\frac{15}{15+5}=\frac{3}{4}$$

- Recall =

$$\frac{15}{15+5}=\frac{3}{4}$$

Hence, our algorithm is 80% accurate, with a precision of 75% and a recall of 75%.

Area Under the Receiver Operating Curve (AUC-ROC)

The area under the receiver operating characteristic (ROC) curve, also known as the AUC-ROC for short, is another widely used metric for evaluating classification machine learning problems. A significant feature of the AUC-ROC metric is that it can be a good metric for evaluating datasets with imbalanced classes.

An imbalanced class is when one of the outcome targets has far more samples than another target. A typical example of this can be seen in a fraud identification dataset where the samples of no fraud will be vastly more than the samples with fraud.

To better understand AUC-ROC, let us derive two (2) relevant formulas from our confusion matrix, and they are the ***True negative rate (TNR)*** (also known as ***specificity***) and the ***False positive rate*** (also known as ***fall-out***).

Specificity is the fraction of results that are ***correctly predicted as negative*** over all the results that are negative, whereas fall-out is the fraction of results that are ***wrongly predicted as positive*** over all the results that are negative. Fall-out is also represented as (1 – specificity).

This is further illustrated in Figure 14-9.

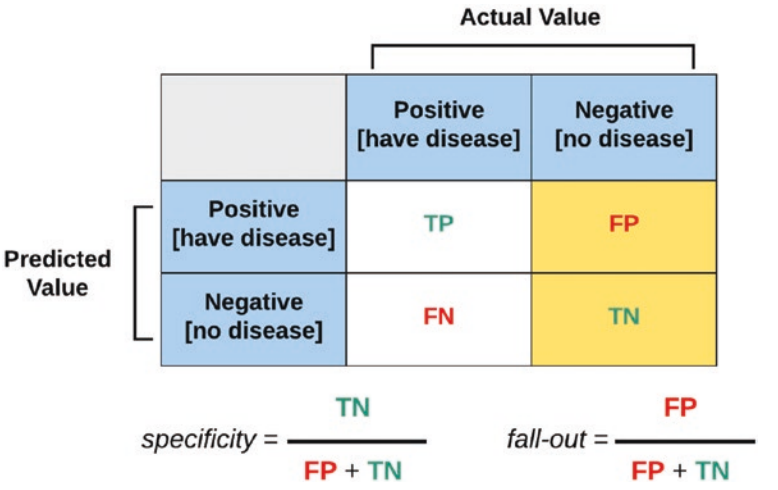

Figure 14-9. *Specificity and fall-out*

The ROC Space

The ROC or receiver operating characteristic space is a 2-D graph that ***plots*** the cumulative probability distribution of the sensitivity (i.e., the probability distribution of making the correct prediction) on the y axis and the cumulative probability distribution of the fall-out (i.e., the probability distribution of a false alarm) on the *x* axis.

A few notable details about the ROC space is that

- The area of the square space is 1 because the x and y axes range from 0 to 1, respectively.
- The diagonal line drawn from point ($x = 0$, $y = 0$) to ($x = 1$, $y = 1$) represented pure chance or a random guess. It is also known as the line of no discrimination.

These expressions are further illustrated in Figure 14-10.

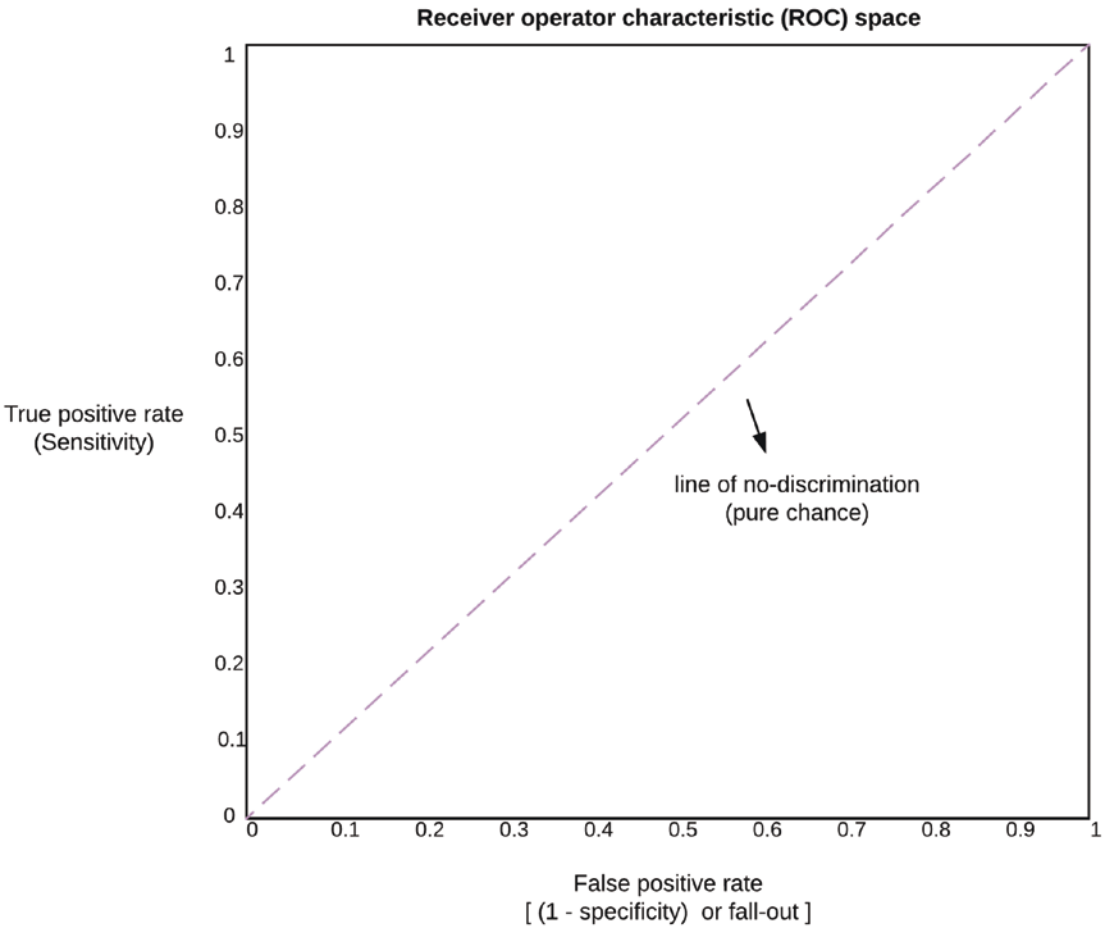

Figure 14-10. *Receiver operating characteristic (ROC) space*

The AUC-ROC Space

The ROC plot as shown in Figure 14-10 looks like a curve. So, the area under the curve, also known as AUC, is the area underneath the ROC curve. AUC provides a single floating-point number that describes the model's performance, and it is interpreted as follows:

- An AUC value below 0.5 indicates that the model's prediction is worse than a random guess of the targets.
- An AUC value closer to 1 signifies a model that is performing very well by generalizing to new examples on the test dataset.

A ROC curve that is closer to the top-left part of the ROC space (i.e., closer to the value 1) indicates that the model has a good classification accuracy.

The AUC-ROC curve is illustrated in Figure 14-11.

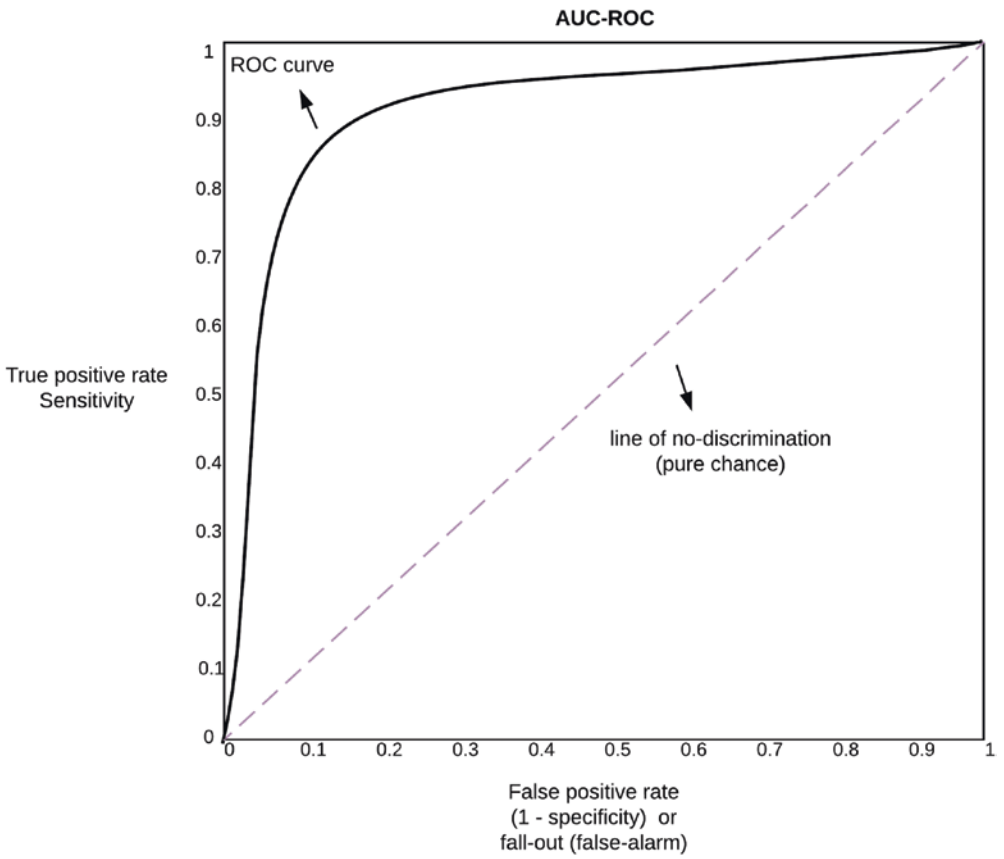

Figure 14-11. *AUC-ROC curve*

Regression Evaluation Metrics

In this section, we'll go through some of the metrics for evaluating regression machine learning tasks.

Root Mean Squared Error (RMSE)

Root mean squared error also known as RMSE for short is an important evaluation metric in supervised machine learning for regression problems. RMSE computes the error difference between the original value of the target feature and the value predicted by the learning algorithm. RMSE is formally represented by the following formula:

$$RMSE = \sqrt{\frac{\sum_{i=1}^{n}\left(y_i - \hat{y}_i\right)^2}{n}}$$

where

- n is the number of samples in the dataset
- y_i is the actual value of the target feature
- $\hat{y}_i$ is the target value predicted by the learning algorithm

Further notes on RMSE:

- Squaring the difference between the actual value and predicted value of the labels $\left(y_i - \hat{y}_i\right)^2$ gives the positive deviation (i.e., the magnitude) between the two numbers.
- Dividing by n gives the average of the sum of magnitudes. The square root ***returns the results in the same unit of measurement*** as the target feature.

An Example of Evaluation with RMSE

Assume we want to predict the price of houses (in thousands of Naira[1]), and we have the following dataset (see Figure 14-12).

[1]Naira is the currency of Nigeria. It is also symbolized by the code NGN and the sign ₦.

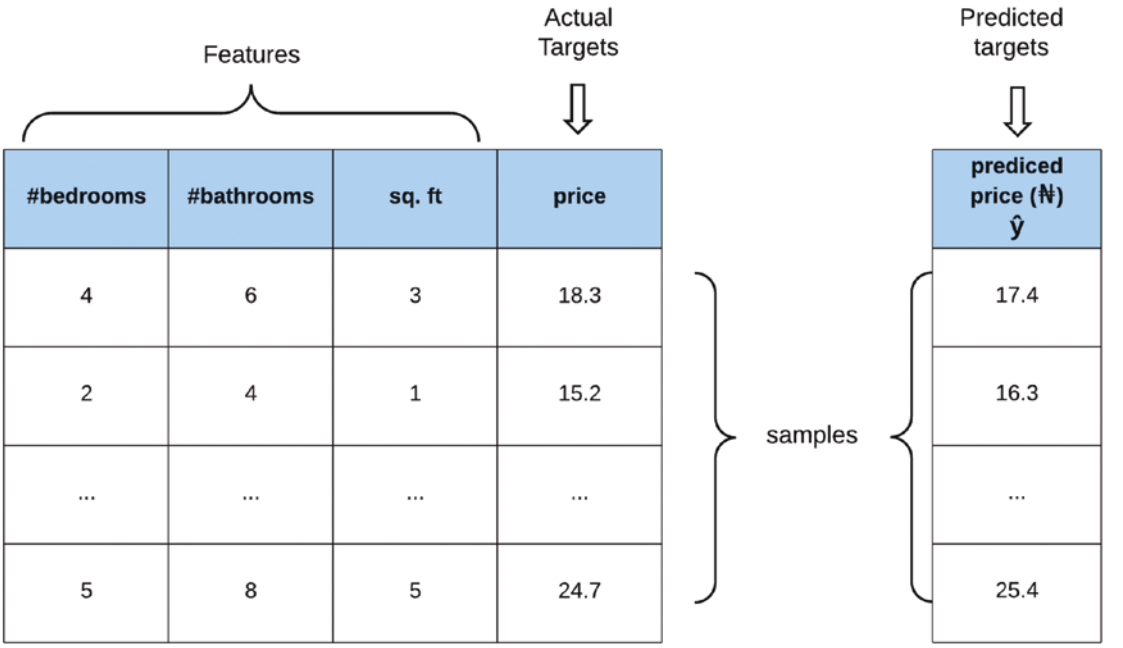

Figure 14-12. *RMSE illustration*

From the formula given, we calculate the RMSE as follows:

$$RMSE = \sqrt{\frac{(18.3-17.4)^2+(15.2-16.3)^2+\ldots+(24.7-25.4)^2}{3}}$$

$$= \sqrt{\frac{(0.9)^2+(-1.1)^2+\ldots+(-0.7)^2}{3}} = \sqrt{\frac{0.81+1.21+\ldots+0.49}{3}}$$

$$= \sqrt{\frac{2.51}{3}} = \sqrt{0.83666666666} = \sqrt{MSE} = 0.91469484893$$

The closer the RMSE is to 0, the better the performance of the model. Again, we are most interested in knowing the RMSE on the test data, as this gives us an accurate picture of the performance of our model. In this example, the error difference between the actual price and predicted price of houses made by our learning model is approximately NGN 910 (i.e., $0.91 * 1000$).

Hence, we can calculate the percentage error as

$$\%error = \frac{error\ difference}{mean\ of\ the\ actual\ prices, \underline{y}} = \frac{0.91}{19.4}$$
$$= 0.04690721649 = \left(0.04690721649 * 100\right) \approx 4\%$$

R-squared (R^2)

R-squared, written as R^2, is another regression error metric that gives us a different perspective into the performance of a learned model. R^2 is also known as the ***coefficient of determination***. The goal of R^2 is to tell us how much of the variance or the variability in the target feature, y, is explained or is captured by the model.

Recall that a model has high variance when it has learned closely the underlying structure of the targets in the dataset. ***Of course, we are mostly concerned with the*** R^2 ***metric on test data***. We typically want the R^2 value on test data to be high. It shows that our model generalizes well to new examples.

Interpretation of R^2

R^2 outputs a value between 0 and 1. Values close to 0 show that variability in the responses are not properly captured by the model, while values close to 1 indicate that the model explains the variability in the observed values. R^2 is calculated using the equation

$$R^2 = 1 - \frac{RSS}{TSS}$$

where

- RSS (i.e., the residual sum of squares) captures the error difference (or the variability) between the actual values and the values predicted by the learning algorithm. The formula is

$$RSS = \sum_{i=1}^{n} \left(y_i - \hat{y}_i\right)^2$$

- *TSS* (i.e., the total sum of squares), on the other hand, calculates the variability in the response variable, *y*. So, for each observation in the dataset, we find the squared difference from the mean of all observation, $\underline{y}$. The formula is

$$TSS = \sum_{i=1}^{n} \left(y_i - \underline{y}_i \right)^2$$

- Hence, $\frac{RSS}{TSS}$ gives us a ratio of how much of the variability in the response variable *y* ***is not explained*** by the model.

So, when we say $1 - \frac{RSS}{TSS}$, we reverse the definition to tell us the ratio of the variability in the response variable explained by the model.

An Example of Evaluating the Model Performance with R^2

Using the dataset illustrated in Figure 14-12 and from the formula given earlier, we will calculate R^2 as follows:

$$RSS = \sum_{i=1}^{n} \left(y_i - \hat{y}_i \right)^2 = \left[\left(18.3 - 17.4\right)^2 + \left(15.2 - 16.3\right)^2 + \ldots + \left(24.7 - 25.4\right)^2 \right]$$
$$= \left[\left(0.9\right)^2 + \left(-1.1\right)^2 + \ldots + \left(-0.7\right)^2 \right] = \left[0.81 + 1.21 + \ldots + 0.49\right] = 2.51$$

while for *TSS*, we have that the mean of the response variable price, $\underline{y}$, is

$$\frac{18.3 + 15.2 + 24.7}{3} = 19.4$$

$$TSS = \sum_{i=1}^{n} \left(y_i - \underline{y}_i \right)^2 = \left[\left(18.3 - 19.4\right)^2 + \left(15.2 - 19.4\right)^2 + \ldots + \left(24.7 - 19.4\right)^2 \right]$$
$$= \left[\left(-1.1\right)^2 + \left(-4.2\right)^2 + \ldots + \left(5.3\right)^2 \right] = \left[1.21 + 17.64 + \ldots + 28.09\right] = 46.94$$

Finally,

$$R^2 = 1 - \frac{RSS}{TSS} = 1 - \frac{2.51}{46.94} = 1 - \left(0.05347251811\right) = 0.94652748189$$

The result shows that the model does a good job in capturing the variability in the target feature. ***Of course, we want to have such good performances on the test dataset***.

Resampling Techniques

This section describes another vital concept for evaluating the performance of supervised learning methods. Resampling methods are a set of techniques that involve selecting a subset of the available dataset, training on that data subset, and then using the remainder of the data to evaluate the trained model.

This process involves creating subsets of the available data into a training set and a validation set. The training set is used to train the model, while the validation set will evaluate the performance of the learned model on unseen data. Typically, this process will be carried out repeatedly to get an approximate measure of the training and test errors.

We will examine three techniques for data resampling and also give some examples of when to use a particular technique. The techniques we'll examine are

- The validation set technique (or train-test split)
- The leave-one-out cross-validation (LOOCV) technique
- The *k*-fold cross-validation technique

The Validation Set

The validation set is the simplest approach for data resampling, and it involves randomly dividing the dataset into two parts; these are the training set and the validation set. The division can be into two equal sets if you have a big enough dataset, or it could be a 60/40 or 70/30 split.

After splitting, the model is trained on the training set, and its performance is evaluated on the validation set. This process is summarized in the list as follows:

1. Randomly split the dataset into
   - Training set
   - Validation set
2. Train the model on the training set.