This post will explore using R’s **MLmetrics** to evaluate machine learning models. **MLmetrics** provides several functions to calculate common metrics for ML models, including AUC, precision, recall, accuracy, etc.

**Building an example model**

Firstly, we need to build a model to use as an example. For this post, we’ll be using a dataset on [pulsar stars from Kaggle](https://www.kaggle.com/pavanraj159/predicting-a-pulsar-star). Let’s save the file as “pulsar\_stars.csv”. Each record in the file represents a pulsar star candidate. The goal will be to predict if a record is a pulsar star based upon the attributes available.

To get started, let’s load the packages we’ll need and read in our dataset.

library(MLmetrics)

library(dplyr)

stars = read.csv("pulsar\_stars.csv")

Next, let’s split our data into train vs. test. We’ll do a standard 70/30 split here.

set.seed(0)

train\_indexes = sample(1:nrow(stars), .7 \* nrow(stars))

train\_set <- stars[train\_indexes,]

test\_set <- stars[-train\_indexes,]

Now, let’s build a simple logistic regression model.

train\_set <- data.frame(train\_set %>% select(target\_class), train\_set %>% select(-target\_class))

# build model

model <- glm(formula(train\_set), train\_set, family = "binomial")

**AUC / precision / recall / accuracy**

Let’s calculate a few metrics. One of the most common metrics for classification is calculating AUC, which can be done using **MLMetrics’** AUC function. Intuitively, AUC is a score between 0 and 1 that measures how well a model rank-orders predictions.

# get AUC on test and train set

AUC(test\_pred, test\_set$target\_class) # 0.974172

AUC(train\_pred, train\_set$target\_class) # 0.9773794

As a refresher, here’s a quick overview of precision, recall, and accuracy:

 **Precision:** The true positive rate. If the model predicts there are 10 pulsar stars, and 8 of those 10 actually are pulsars, then the precision would be 8 / 10, or 80%.

 **Recall:**The proportion of the positive labels that are captured with the model. For example, suppose there are 10 pulsar stars in the data and that the model predicts 7 of those to be pulsar stars. That would mean the recall is 7 / 10, or 70%.

 **Accuracy:**Generally the most intuitive of the metrics here. Accuracy is simply the number of correct predictions divided by the total number of predictions.

Notice how each above metric requires whole number inputs. To handle this, we need to set a threshold on our predicted probabilities. One way to do this would be to assign any prediction above 50% as a predicted pulsar star, while any prediction that is less than 50% would get assigned as *not* a pulsar star.

For example, if we pick 0.5 as a threshold, our precision on the test set would be 0.9114219.

Precision(test\_set$target\_class, ifelse(test\_pred >= .5, 1, 0), positive = 1) # 0.9114219

Rather than just picking 0.5, though, we can try to optimize the cutoff we choose. One method of accomplishing this is to choose the threshold that optimizes the F1 Score. F1 Score is defined as the [harmonic mean](https://en.wikipedia.org/wiki/Harmonic_mean) between precision and recall

Below, we calculate the F1 Score for each threshold 0.01, 0.02, 0.03,…0.99. The threshold that gives the optimal cutoff (optimal F1 Score) is .32, or 32%.

f1\_scores <- sapply(seq(0.01, 0.99, .01), function(thresh) F1\_Score(train\_set$target\_class, ifelse(train\_pred >= thresh, 1, 0), positive = 1))

which.max(f1\_scores) # 32

Using this cutoff, we can calculate precision, recall, and accuracy.

Precision(test\_set$target\_class, ifelse(test\_pred >= .32, 1, 0), positive = 1)

Recall(test\_set$target\_class, ifelse(test\_pred >= .32, 1, 0), positive = 1)

Accuracy(ifelse(test\_pred >= .32, 1, 0), test\_set$target\_class)

In general, there will be a trade-off between precision and recall, so the selection of a threshold may also vary depending on which of those metrics is more valued. Optimizing based off F1 Score is a good way to try to optimize the threshold based off both precision and recall.

**Gini**

Another metric that can be used in evaluating classification models is the **Gini** coefficient. Gini is calculated as 2 \* AUC – 1. Thus, we get 0.974172 \* 2 – 1 = 0.948344.

Gini(test\_pred, test\_set$target\_class) # 0.948344

**Other metrics**

**MLmetrics** also has functions for non-classification metrics as well, such as RMSE and RAE.