Is your Classification Model making lucky guesses?

Shaheen Gauher, PhD, Data Scientist at Microsoft

At the heart of a classification model is the ability to assign a class to an object based on its description or features. When we build a classification model, often we have to prove that the model we built is significantly better than random guessing. How do we know if our machine learning model performs better than a classifier built by assigning labels or classes arbitrarily (through random guess, weighted guess etc.)? I will call the latter, non-machine learning classifiers as these do not learn from the data. A machine learning classifier should be smarter and not just making lucky guesses. At the least it should do a better job at detecting the difference between different classes and should have a better accuracy than the latter. In the sections below, I will show three different ways to build a non-machine learning classifier and compute their accuracy. The purpose is to establish some baseline metrics against which we can evaluate our classification model. Lastly, I will discuss Kappa statistics which compares the accuracy of a classifier to the accuracy that could be achieved purely by chance.

I will present here a detailed description of how to build these classifiers based on the knowledge of the distribution of the data. I algebraically derive the performance metrics of these non-machine classifiers. These are our baseline metrics. I also show the MRS code to build a multiclass classification model using iris data set and compare the accuracy of this model to various baseline metrics.

In the examples below, I will use data with population size n. The data is divided into two groups, with $\mathrm{x}\%$ of the rows belonging to one class (labeled positive or P) and $(1\text{-}\mathrm{x})\%$ belonging to another class (labeled negative or N). This is the ground truth.

$$Population\ Size = n$$

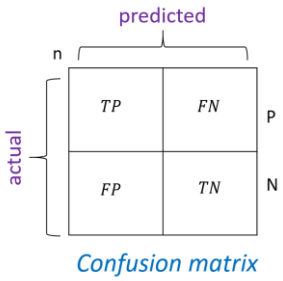
$$Fraction\ of\ instances\ labelled\ positive = x$$

$$Fraction\ of\ instances\ labelled\ negative = (1-x)n$$

$$Number\ of\ instances\ labelelled\ positive(P) = xn$$

$$Number\ of\ instances\ labelled\ negative(N) = (1-x)n$$

The confusion matrix with rows reflecting the ground truth and columns reflecting the machine learning classifier classifications looks like:



Confusion matrix

Now

$P = TP + FN = xn$ (1)

$N = FP + TN = (1 - x)n$ (2)

$n = TP + FP + TN + FN = P + N$ (3)

where

$TP = True\ Positive$

$TN = True\ Negative$

$FP = False\ Positive$

$FN = False\ Negative$

1) Random Guess Classifier

Let us build a model by randomly assigning half of the labels to positive and half to negative. This random guessing is actually just another classifier. What is the accuracy of this classifier?

Since this model is labelling half the data as positive and other half as negative:

$TP + FP = \frac{n}{2}$ (4)

$TN + FN = \frac{n}{2}$ (5)

Also the random model will assign half of the actual positive labels as positive and half of actual negative labels as negative:

$TP = FN$ (6)

$FP = TN$ (7)

Based on above equations and some algebra:

$TP = P - FN$

$= xn - FN$ .. from (1)

$= xn - TP$ .. from (6)

$= \frac{xn}{2}$

$TN = N - FP$

$= N - TN$ .. from (7)

$= \frac{N}{2}$

$= \frac{(1-x)n}{2}$ .. from (2)

$Accuracy = \frac{TP+TN}{n} = \frac{\frac{xn}{2}+\frac{(1-x)n}{2}}{n} = \frac{\frac{n}{2}}{n} = 0.5$ (8)

$Recall = \frac{TP}{TP+FN} = \frac{TP}{P} = \frac{\frac{xn}{2}}{xn} = 0.5$ (9)

$Precision = \frac{TP}{TP+FP} = \frac{\frac{xn}{2}}{\frac{n}{2}} = x$ (10)

For a multiclass classification with k classes with x_i being the fraction of instances belonging to class i (i = 1 to k), it can similarly be shown that the accuracy would by $\frac{1}{k}$. The precision for a class i would be x_i, the fraction of instances in the data with class i. Recall for a class will be equal to $\frac{1}{k}$.

In the language of probability, the accuracy is simply the probability of selecting a class which for two classes (binary classification) is 0.5 and for k classes will be $\frac{1}{k}$.

The MRS code to compute the above using iris data set can be found here.

2) Weighted Guess Classifier

Let us build a model by assigning the labels as positive and negative based on a weighted guess instead of randomly. The weights are based on the distribution of classes in the data (positive label is x % of the data in this example). This weighted guessing is also just another classifier. What is the accuracy of this classifier where the classes are assigned as follows: x% of actual positives are assigned as positive by this model (1.e. TP = x % of P) and (1-x) % of actual negatives are assigned as negatives (i.e. TN = (1-x) % of N).

$TP = xP$ (11)

$TN = (1 - x)N$ (12)

Now

$FP = N - TN$

$= N - (1 - x)N$ .. from (12)

$$= xN$$

$$FN = P - TP$$

$$= P - xP \qquad \text{.. form (11)}$$

$$= P(1 - x)$$

$$Accuracy = \frac{TP+TN}{n}$$

$$= \frac{xP+(1-x)N}{n}$$

$$= \frac{xxn+(1-x)(1-x)n}{n} \qquad \text{.. from (1) and (2)}$$

$$= x^2 + (1 - x)^2 \qquad (13)$$

$$Recall = \frac{TP}{P} = \frac{xP}{P} = x \qquad (14)$$

$$Precision = \frac{TP}{(TP+FP)} = \frac{xP}{xP+xN} = \frac{P}{P+N} = \frac{P}{n} = x \qquad (15)$$

For a multiclass classification with k classes with x_i being the fraction of instances belonging to class i (i = 1 to k), it can similarly be shown that

$$Accuracy = \sum_{i=1}^{k} x_i^2$$

Precision and Recall for a class would equal to x_i, the fraction of instances in the data with the class i.

In the language of probability, $\frac{xn}{n}$ or x is the probability of a label being positive in the data (for negative label, the probability is $\frac{(1-x)n}{n}$ or $(1-x)$). If there are more negative instances in the data, the model has a higher probability of assigning an instance as negative. The probability of assigning true positives by the model will be $x*x$ (for true negative it is $(1-x)^2$. The accuracy is simply the sum of these two probabilities.

The MRS code to compute the above using iris data set can be found here.

3) Majority Class Classifier

Let us build a model by assigning all the labels to negative assuming that is the majority class. This majority class labelling can also be just another classifier. What is the accuracy of this classifier? The accuracy of this classifier is also called ***No Information Rate (NIR)*** and is reported by the caret package in R. ***Null Error Rate*** is (1 – Accuracy or the No Information Rate) and tells how often we would be wrong if we always predicted the majority class. Since all labels are assigned as negative we get

$$TN + FN = n \qquad (16)$$

$$TP = 0 \quad (17)$$

$$FP = 0 \quad (18)$$

$$TN = N - FP$$
$$= N$$
$$= (1 - x)n$$

$$FN = P - TP$$
$$= P$$
$$= xn$$

$$Accuracy = \frac{TP+TN}{n} = \frac{TN}{n} = \frac{(1-x)n}{n} = (1 - x) \quad (19)$$

$$Recall = \frac{TN}{N} = \frac{(1-x)n}{(1-x)n} = 1 \quad (20)$$

$$Precision = \frac{TN}{TN+FN} = \frac{TN}{n} = \frac{(1-x)n}{n} = (1 - x) \quad (21)$$

For a multiclass classification with k classes with x_i being the fraction of instances belonging to class i (i = 1 to k), it can similarly be shown that, Accuracy will be equal to (1-x_i), the fraction of instances belonging to the majority class (assumed negative here). Recall will be 1 for the majority class and 0 for all other classes. Precision will be equal to the fraction of instances belonging to the majority class for the majority class and 0 for all other classes.

In the language of probability (assuming that the negative label is majority), the probability of assigning a positive label to an instance by the model will be zero and the probability of assigning a negative label to an instance will be 1.

The MRS code to compute the above using iris data set can be found here.

You can download the entire MRS code here.

Kappa Statistics

The accuracy of a machine learning classifier is, to put it simply, a measure of agreement between the actual and predicted classes. Kappa Statistic compares the accuracy of the machine learning classifier to the accuracy of a random system (i.e. how much accuracy we expect to achieve purely by chance). I will call the accuracy of the random system, the Expected Accuracy. If the machine learning classifier yields a total accuracy of 85% when the Expected Accuracy was 80%, it is not as great as when the Expected Accuracy is say 40%. To compute Expected accuracy, we first compute the expected frequency of agreements by chance between prediction and actual for each class. The sum of the expected frequencies of

agreement by chance for each class gives the Expected Accuracy. The Expected Accuracy is then used to compute Kappa as.

$$Kappa = \frac{TotalAccuracy - ExpectedAccuracy}{1 - ExpectedAccuracy} \quad (22)$$

where

$$Expected\ Accuracy = \frac{ActualNegative(N) * PredictedNegative + ActualPositive(P) * PredictedPositive}{n * n}$$

$$ExpectedAccuracy = \frac{(TN+FP)*(TN+FN)+(FN+TP)*(FP+TP)}{n*n} \quad (23)$$

The MRS code to compute Kappa can be found here.

Interpreting Kappa

Kappa statistics is normalized statistic with values between 0 and 1. Larger values of Kappa indicate better reliability of the classifier. A perfect agreement between actual and predicted would yield a kappa of 1, where a chance agreement would equate to 0. Compared to simple accuracy, Kappa statistics is especially helpful when class distributions are skewed as it controls for the skewedness through the Expected accuracy. For the same classification task, Kappa statistics can be used more reliably to compare between different models. There is no universally accepted Kappa scale but a commonly cited scale for Kappa form Landis and Koch is as follows

Kappa	Agreement
< 0	No agreement
0.01 – 0.20	Slight agreement
0.21 – 0.40	Fair agreement
0.41 – 0.60	Moderate agreement
0.61 – 0.80	Substantial agreement
0.81-1	Almost perfect agreement

Landis, J.R.; Koch, G.G. (1977). "The measurement of observer agreement for categorical data". Biometrics 33 (1): 159–174. doi:10.2307/2529310. JSTOR 2529310. PMID 843571.