Confusion Matrix

Machine Learning - Section 2.1.2

Marc Omar Haddad

20 January, 2020

We previously defined a prediction rule that predicts `male` any height above 64 inches. However, given the fact that `female` average height is about 65 inches, there appears to be a conflict with our rule: if the average `female` height is 65 inches, why does our rule tell us to predict `male` for those who are 65 inches tall?

> Overall accuracy can be a deceptive measure.
> -*Dr. Rafael Irizarry*

We can see this by constructing a **Confusion Matrix**: A tabulation of *prediction* and *actual value*.

```
table(predicted = y_hat, actual = test_set$sex)
```

```
##          actual
## predicted Female Male
##    Female     50   27
##    Male       69  379
```

The table above is read like so:

- Subjects that were *actually* `female` **and** *predicted* to be `female`: 50
- Subjects that were *actually* `male` **but** *predicted* to be `female`: 27
- Subjects that were *actually* `female` **but** *predicted* to be `male`: 69
- Subjects that were *actually* `male` **and** *predicted* to be `male`: 379

Looking at the above values closely, a problem emerges: `male`'s are *over represented* in our data set. Computing the accuracy separately for each sex reflects as much:

```
test_set %>%
  mutate(y_hat = y_hat) %>%
  group_by(sex) %>%
  summarize(accuracy = mean(y_hat == sex))
```

sex	accuracy
Female	0.420
Male	0.933

Our algorithm has very high accuracy when predicting `male`: 93.3%, but extremely low accuracy when predicting `female`: 42.0%. **Too many `female`'s are predicted to be `male`**.

Our high overall accuracy despite our low `female` accuracy is due to **Prevalence**: There are more `male`'s in our data set than `female`'s.

```
prev = mean(y == "Male")
prev
```

```
## [1] 0.773
```

What this means is that the *incorrect* predictions of `female`'s is **outweighed** by correctly predicting more `male`'s. This is a significant problem in Machine Learning: If our training data is biased, our algorithm will be biased. Therefore, we need to look at metrics other than overall accuracy, that are robust to prevalence, when evaluating a Machine Learning algorithm.

Derived Metrics from the Confusion Matrix: Sensitivity and Specificity

A general improvement to only studying overall accuracy is to study **Sensitivity** and **Specificity** separately. These two metrics can be defined with a binary outcome. When the outcomes are **categorical**, we can define these metrics for a *specific* category. For example, when predicting digits, we can calculate the Sensitivity and Specificity of correctly predicting "2" as opposed to some other digit. By selecting "2" we've specified a category of interest.

When $Y = 1$, we will define these outcomes as **positive outcomes**. When $Y = 0$, we will define these outcomes as **negative outcomes**.

Sensitivity is defined as the ability of an algorithm to **predict a positive outcome**, $\hat{Y} = 1$, when the **actual outcome is positive**, $Y = 1$. However, because an algorithm that predicts all Y's to be positive (i.e. an algorithm has $\hat{Y} = 1$, no matter what) has **perfect sensitivity**, Sensitivity alone is not adequate when evaluating algorithms.

Specificity is defined as the ability of an algorithm to **not predict a positive outcome**, $\hat{Y} = 0$, when the **actual outcome is not positive**, $Y = 0$. We can summarize like so:

$$\text{High Sensitivity}: \; Y = 1 \underset{implies}{\Rightarrow} \hat{Y} = 1$$

$$\text{High Specificity}: \; Y = 0 \underset{implies}{\Rightarrow} \hat{Y} = 0$$

There is a second way to define **Specificity**: The proportion of **positive predictions** that are **actually positive**(i.e. (correct positive predictions) / (all positive predictions)). In this case high Specificity is defined as:

$$\text{High Specificity}: \; \hat{Y} = 1 \underset{implies}{\Rightarrow} Y = 1$$

To differentiate between these metrics, each of the four entries in the confusion matrix has a unique name.

	Actually.Positive	Actually.Negative
Predicted +	True Positives(TP)	False Positives(FP)
Predicted -	False Negatives(FN)	True Negatives(TN)

- When an **outcome** is **positive** and was **predicted** to be **positive**: True Positives (TP)
- When an **outcome** is **negative** but was **predicted** to be **positive**: False Positives (FP)
- When an **outcome** is **positive** but was **predicted** to be **negative**: False Negatives (FN)
- When an **outcome** is **negative** and was **predicted** to be **negative**: True Negatives (TN)

With these definitions, we can accurately quantify Sensitivity and Specificity:

$$\text{Sensitivity}_1 = \frac{TP}{(TP + FN)}$$

$$\text{Specificity}_2 = \frac{TN}{(TN + FP)}$$

There is another way of quantifying Specificity:

$$\text{Specificity}_3 = \frac{TP}{(TP + FP)}$$

[1] This formulation of Sensitivity is also known as the **True Positive Rate** (**TPR**) or **Recall**.

[2] This formulation of Specificity is also known as the **True Negative Rate** (**TNR**).

[3] This formulation of Specificity is also known as the **Positive Predictive Value** (**PPV**) or **Precision**.

To help in understanding the above, keep in mind that:

$(TP + FN)$ = All positive outcomes
$(TN + FP)$ = All negative outcomes
$(TP + FP)$ = All positive predictions

Note that unlike the True Positive Rate and the True Negative Rate, **Precision** is affected by **Prevalence**. The higher the Prevalence, the higher the Precision.

Summary Table

Measure	Name_1	Name_2	Calculation	As_Probability
Sensitivity	TPR	Recall	TP / (TP+FN)	Pr(Y_hat = 1 | Y = 1)
Specificity	TNR	1 - FPR	TN / (TN+FP)	Pr(Y_hat = 0 | Y = 0)
Specificity	PPV	Precision	TP / (TP+FP)	Pr(Y = 1 | Y_hat = 1)

TPR = True Positive Rate
TNR = True Negative Rate
PPV = Positive Predictive Value
FPR = False Positive Rate (Not covered in this lecture)

`confusionMatrix()`

The `confusionMatrix` function in the `caret` package computes all of the aforementioned metrics once we define what our algorithm should consider to be positive.
The function expects factors as inputs. The first factor level is considered to represent our "positive" outcome $Y = 1$. In our example, `female` is the first level.

The usage of `confusionMatrix` is as follows:

```r
confusionMatrix(data = y_hat, reference = test_set$sex)
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction Female Male
##     Female     50   27
##     Male       69  379
##
##                Accuracy : 0.817
##                  95% CI : (0.781, 0.849)
##     No Information Rate : 0.773
##     P-Value [Acc > NIR] : 0.00835
##
##                   Kappa : 0.404
##
##  Mcnemar's Test P-Value : 2.86e-05
##
##             Sensitivity : 0.4202
##             Specificity : 0.9335
##          Pos Pred Value : 0.6494
##          Neg Pred Value : 0.8460
##              Prevalence : 0.2267
##          Detection Rate : 0.0952
##    Detection Prevalence : 0.1467
##       Balanced Accuracy : 0.6768
##
##        'Positive' Class : Female
##
```