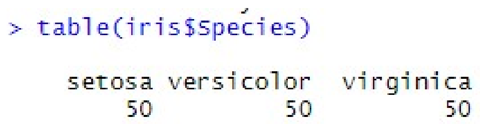
Let’s consider the above example, where we had class A with 90 observations and class B with 10 observations. If we predict the entire data set as class A, we will achieve an accuracy of 90% which seems really not bad for a classification model. But, in reality this model is very poor. To expand on this further, if we consider a quality control process and the model predicts everything as good, then we end up getting a lot of warranty claims.

So, what metrics should we be looking at to determine how our model is performing? Let’s take an example to do a walk through.

Example 1: Balanced Data Set

For this we will consider iris data set. Since, the “iris” data set is already balanced as shown in the below figure, the model is bound to perform well.



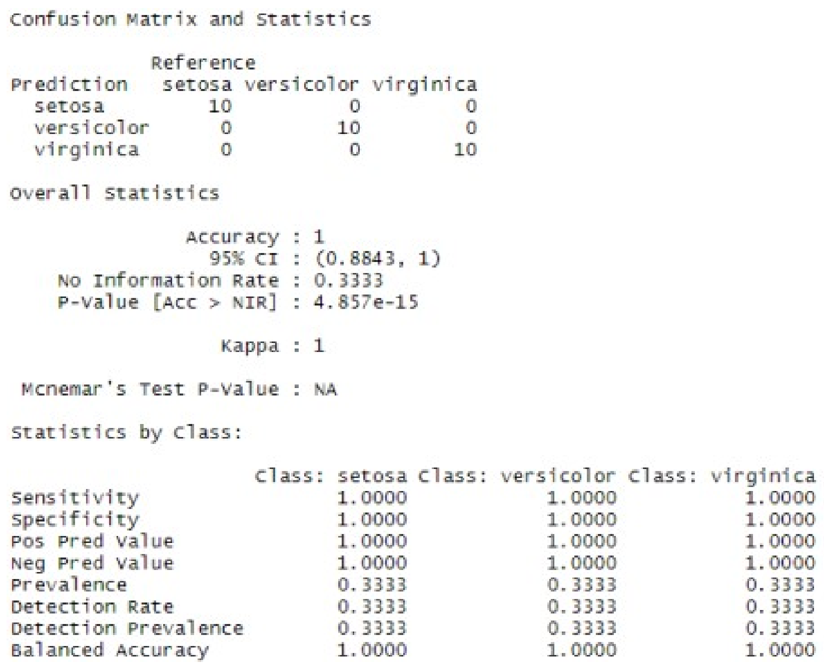
```
> table(iris$Species)

    setosa versicolor  virginica 
        50         50         50 
```

```
# load the libraries
library(e1071)
library(caret)

# load iris dataset
data("iris")

# create a sample for test and train
sample = createDataPartition(iris$Species, p = 0.8)

# create test and train set
train = iris[sample$Resample1,]
test = iris[-sample$Resample1,]

# build a svm model
svm_model = svm(x = train[,1:4], y = train[,5], type = "C-classification")

# perform perdiction on test data set
predictions = predict(svm_model, test[,1:4])

# print confusion matrix
confusionMatrix(test$Species, predictions)
```

In the below confusion matrix, we could notice that all the classes were predicted accurately and hence the accuracy is 100%. No Information Rate (NIR) can be explained as if we randomly guess which class the test observations belong to, forms NIR. Ideally, NIR should be less than Accuracy. Here, NIR is 0.33 which is less than accuracy indicating a good model. Also, we notice that NIR = 0.33 indicating equal balanced classes.

Next metric that we want to look at is balanced accuracy. The balanced accuracy in binary and multi-class classification problems to deal with imbalanced data sets. It is defined as the average of recall obtained on each class. The best value is 1 and the worst value is 0. From the below result, we notice that for all 3 classes Balanced Accuracy is 100% indicating a well balanced model.

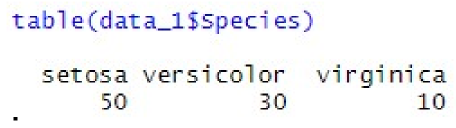
Finally, looking at detection rate we notice that detecting setosa, versicolor or virginica are all same at 0.33.



```
Confusion Matrix and Statistics

            Reference
Prediction   setosa versicolor virginica
  setosa         10          0         0
  versicolor      0         10         0
  virginica       0          0        10

Overall Statistics

               Accuracy : 1
                 95% CI : (0.8843, 1)
    No Information Rate : 0.3333
    P-Value [Acc > NIR] : 4.857e-15

                  Kappa : 1

 Mcnemar's Test P-Value : NA

Statistics by Class:

                     Class: setosa Class: versicolor Class: virginica
Sensitivity                 1.0000            1.0000           1.0000
Specificity                 1.0000            1.0000           1.0000
Pos Pred Value              1.0000            1.0000           1.0000
Neg Pred Value              1.0000            1.0000           1.0000
Prevalence                  0.3333            0.3333           0.3333
Detection Rate              0.3333            0.3333           0.3333
Detection Prevalence        0.3333            0.3333           0.3333
Balanced Accuracy           1.0000            1.0000           1.0000
```

Example 2: Unbalanced Data Set

For the next example, we will take the same iris data set and create an unbalanced data set as shown below.



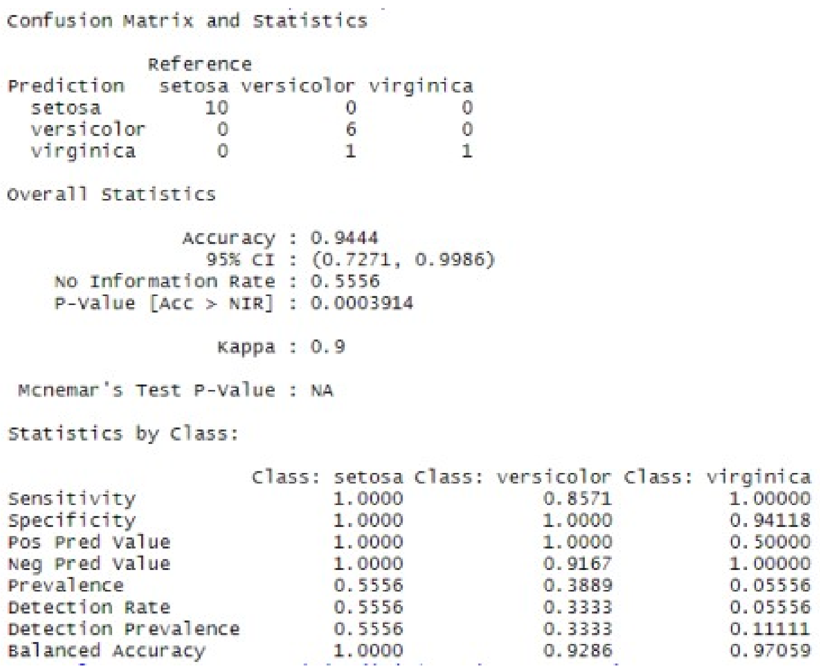
```
table(data_1$Species)

    setosa versicolor  virginica
        50         30         10
```

```
# create an unbalanced data set
data_1 = rbind(iris[1:50,], iris[51:80, ], iris[101:110, ])

# create a sample for test and train
sample = createDataPartition(data_1$Species, p = 0.8)

# create test and train set
train = data_1[sample$Resample1,]
test = data_1[-sample$Resample1,]

# build a svm model
svm_model = svm(x = train[,1:4], y = train[,5], type = "C-classification")

# perform perdiction on test data set
predictions = predict(svm_model, test[,1:4])

# print confusion matrix
confusionMatrix(test$Species, predictions)
```

In the below results, we see that the accuracy is 94% which is not bad for a result. NIR is 0.555, which is indeed less than accuracy. But NIR is significantly higher than the previous example indicating randomly assigning classes for a unbalanced data set would generate higher accuracy than a balanced data set.

Next, we look at balanced accuracy where, class setosa has 1 indicating a perfectly balanced and versicolor and virginica are less than 1.

Finally, looking at detection rate we notice that detecting setosa is higher than versicolor or virginica. Ideally, detection rate for each of the classes should be close to each other or same.



```
Confusion Matrix and Statistics

            Reference
Prediction   setosa versicolor virginica
  setosa         10          0         0
  versicolor      0          6         0
  virginica       0          1         1

Overall Statistics

               Accuracy : 0.9444
                 95% CI : (0.7271, 0.9986)
    No Information Rate : 0.5556
    P-Value [Acc > NIR] : 0.0003914

                  Kappa : 0.9

 Mcnemar's Test P-Value : NA

Statistics by Class:

                     Class: setosa Class: versicolor Class: virginica
Sensitivity                 1.0000            0.8571          1.00000
Specificity                 1.0000            1.0000          0.94118
Pos Pred Value              1.0000            1.0000          0.50000
Neg Pred Value              1.0000            0.9167          1.00000
Prevalence                  0.5556            0.3889          0.05556
Detection Rate              0.5556            0.3333          0.05556
Detection Prevalence        0.5556            0.3333          0.11111
Balanced Accuracy           1.0000            0.9286          0.97059
```

From the above examples, we notice that having a balanced data set for a model would generate higher accuracy models, higher balanced accuracy and balanced detection rate. Hence, its important to have a balanced data set for a classification model.

Let me know in the comments below if ever deal with unbalanced data sets and how you deal with unbalanced data set.