Spam or Non-spam Classifier

Anna Assouline
Chelomo Lubliner
Steve Abecassis

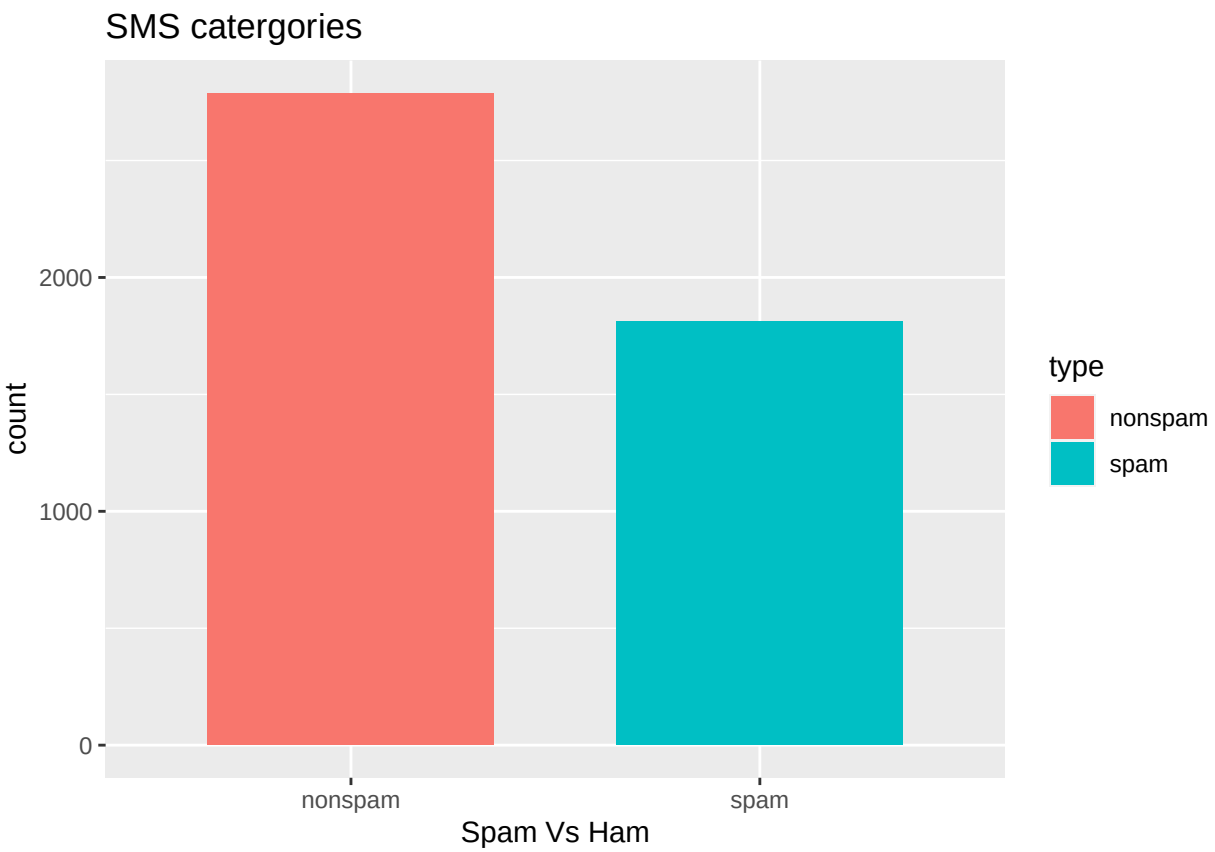
Load the Dataset:

```
df <- read.csv('/Users/steveabecassis/Desktop/spam.csv')
head(df , 3)
```

```
##   make address  all num3d  our over remove internet order mail receive will
## 1 0.00    0.64 0.64     0 0.32 0.00   0.00     0.00  0.00 0.00    0.00 0.64
## 2 0.21    0.28 0.50     0 0.14 0.28   0.21     0.07  0.00 0.94    0.21 0.79
## 3 0.06    0.00 0.71     0 1.23 0.19   0.19     0.12  0.64 0.25    0.38 0.45
##   people report addresses free business email  you credit your font num000
## 1   0.00   0.00      0.00 0.32     0.00  1.29 1.93   0.00 0.96    0   0.00
## 2   0.65   0.21      0.14 0.14     0.07  0.28 3.47   0.00 1.59    0   0.43
## 3   0.12   0.00      1.75 0.06     0.06  1.03 1.36   0.32 0.51    0   1.16
##   money hp hpl george num650 lab labs telnet num857 data num415 num85
## 1  0.00  0   0      0      0   0    0      0      0    0      0     0
## 2  0.43  0   0      0      0   0    0      0      0    0      0     0
## 3  0.06  0   0      0      0   0    0      0      0    0      0     0
##   technology num1999 parts pm direct cs meeting original project   re  edu
## 1          0    0.00     0  0   0.00  0       0     0.00       0 0.00 0.00
## 2          0    0.07     0  0   0.00  0       0     0.00       0 0.00 0.00
## 3          0    0.00     0  0   0.06  0       0     0.12       0 0.06 0.06
##   table conference charSemicolon charRoundbracket charSquarebracket
## 1     0          0          0.00           0.000                 0
## 2     0          0          0.00           0.132                 0
## 3     0          0          0.01           0.143                 0
##   charExclamation charDollar charHash capitalAve capitalLong capitalTotal type
## 1           0.778      0.000    0.000      3.756          61          278 spam
## 2           0.372      0.180    0.048      5.114         101         1028 spam
## 3           0.276      0.184    0.010      9.821         485         2259 spam
```

Present the distribution of Spam and nonspam:

```
library(ggplot2)
fig1 <- ggplot(df )+geom_bar(aes(factor(x= type), fill = type ), width = 0.7  )+ xlab("Spam Vs Ham")+gg
fig1
```



Replace spam by 1 and nonspam by 0 for building the classification models :

```
library(dplyr)
df <- df %>%
          mutate(type = ifelse(type == "spam",1,0))
```

Split the Dataset into Train and Test Set. We will train the model on the train set and verify the model performance on the test set.

```
set.seed(99)

# Create Training Set
X  <- df %>%
          select(c(1:57))

y <-  df %>%
          select(58)

dt = sort(sample(nrow(df), nrow(df)*.8))
train<-df[dt,]
```

```
test<-df[-dt,]

X_train  <- train %>%
              select(c(1:57))
X_test  <- test %>%
              select(c(1:57))
y_train <-  train %>%
              select(58)
y_test  <-  test %>%
              select(58)

nrow(df)
```

```
## [1] 4601
```

```
nrow(train)
```

```
## [1] 3680
```

```
nrow(test)
```

```
## [1] 921
```

In the precedent step we split the Dataset into train and test set, keeping the original ratio between spam and nonspam. Let verify it :

```
#percentage total
table(df['type'])/nrow(df)
```

```
##
##         0         1
## 0.6059552 0.3940448
```

```
#percentage train
table(y_train)/nrow(y_train)
```

```
## y_train
##         0         1
## 0.6119565 0.3880435
```

```
#percentage test
table(y_test)/nrow(y_test)
```

```
## y_test
##         0         1
## 0.5819761 0.4180239
```

As we can see we kept the original ratio.

Function that will help us to present the models performance . We will focus on the confusion matrix , the recall and the precision.

```
draw_confusion_matrix <- function(cm) {

  layout(matrix(c(1,1,2)))
  par(mar=c(2,2,2,2))
  plot(c(100, 345), c(300, 450), type = "n", xlab="", ylab="", xaxt='n', yaxt='n')
  title('CONFUSION MATRIX', cex.main=2)

  # create the matrix
  rect(150, 430, 240, 370, col='#3F97D0')
  text(195, 435, 'nonspam', cex=1.2)
  rect(250, 430, 340, 370, col='#F7AD50')
  text(295, 435, 'spam', cex=1.2)
  text(125, 370, 'Predicted', cex=1.3, srt=90, font=2)
  text(245, 450, 'Actual', cex=1.3, font=2)
  rect(150, 305, 240, 365, col='#F7AD50')
  rect(250, 305, 340, 365, col='#3F97D0')
  text(140, 400, 'nonspam', cex=1.2, srt=90)
  text(140, 335, 'spam', cex=1.2, srt=90)

  # add in the cm results
  res <- as.numeric(cm$table)
  text(195, 400, res[1], cex=1.6, font=2, col='white')
  text(195, 335, res[2], cex=1.6, font=2, col='white')
  text(295, 400, res[3], cex=1.6, font=2, col='white')
  text(295, 335, res[4], cex=1.6, font=2, col='white')

  # add in the specifics
  plot(c(100, 0), c(100, 0), type = "n", xlab="", ylab="", main = "DETAILS", xaxt='n', yaxt='n')
  text(10, 85, names(cm$byClass[1]), cex=1.2, font=2)
  text(10, 70, round(as.numeric(cm$byClass[1]), 3), cex=1.2)
  text(30, 85, names(cm$byClass[2]), cex=1.2, font=2)
  text(30, 70, round(as.numeric(cm$byClass[2]), 3), cex=1.2)
  text(50, 85, names(cm$byClass[5]), cex=1.2, font=2)
  text(50, 70, round(as.numeric(cm$byClass[5]), 3), cex=1.2)
  text(70, 85, names(cm$byClass[6]), cex=1.2, font=2)
  text(70, 70, round(as.numeric(cm$byClass[6]), 3), cex=1.2)
  text(90, 85, names(cm$byClass[7]), cex=1.2, font=2)
  text(90, 70, round(as.numeric(cm$byClass[7]), 3), cex=1.2)

  # add in the accuracy information
  text(30, 35, names(cm$overall[1]), cex=1.5, font=2)
  text(30, 20, round(as.numeric(cm$overall[1]), 3), cex=1.4)
  text(70, 35, names(cm$overall[2]), cex=1.5, font=2)
  text(70, 20, round(as.numeric(cm$overall[2]), 3), cex=1.4)
}
```

We build a classification model on the train set using logistic regression model . After we training the model on the train set , we check the model performance on the test set and presents the result using our function draw_confusion_matrix.

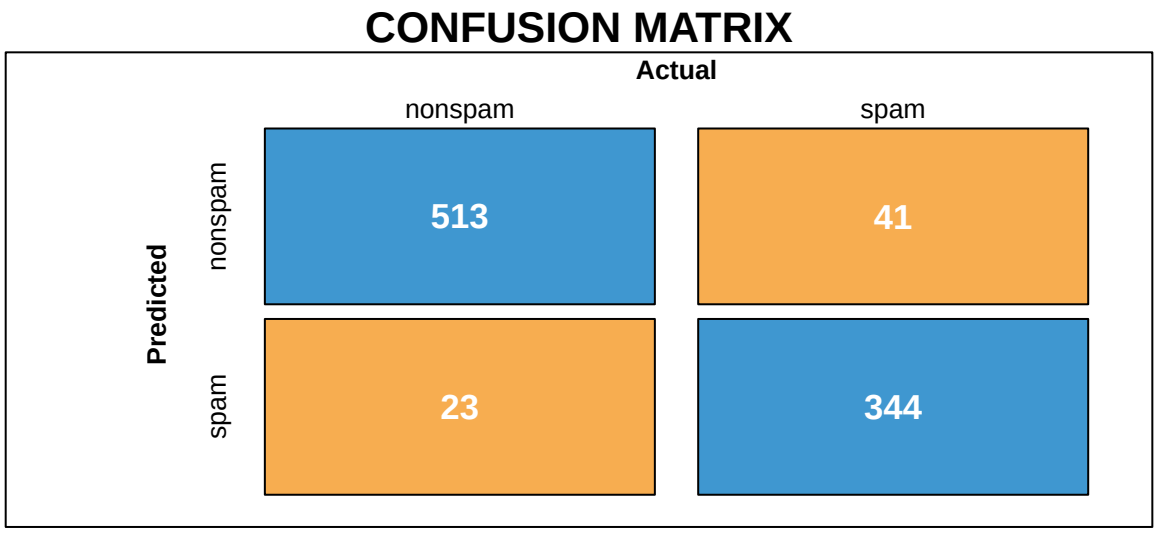
```
library(caret)

spam_classifier <- glm(type  ~ ., data= df, family=binomial(link="logit"))
```

```
## Warning: glm.fit: fitted probabilities numerically 0 or 1 occurred
```

```
pred <- predict(spam_classifier,newdata= X_test,type='response')
pred <- ifelse(pred > 0.5,1,0)

cm <- confusionMatrix(data = as.factor(pred) , reference = as.factor(y_test$type))
draw_confusion_matrix(cm)
```



We can see on the confusion matrix that the model give 513 prediction of nonspam when it was a non spam and 344 spam predictions when is was a spam. (In blue it's a correct prediction). We can see that the model give 23 predictions of spam when it was a nonspam. We can see that the model give 41 predictions of nonspam when it was a spam. We have a precision of 92.6%. It's mean that when we predict that it was a spam , it was a correct prediction in 92.6% of cases. We have a recall of 95.7% it's mean that the model detect 95.7% of the spams.

The build-in function summary can help us to explain the model :

```
print(summary(spam_classifier))
```

```
##
## Call:
## glm(formula = type ~ ., family = binomial(link = "logit"), data = df)
##
## Deviance Residuals:
##     Min       1Q   Median       3Q      Max
## -4.127  -0.203   0.000   0.114   5.364
##
## Coefficients:
```

```
##                     Estimate Std. Error z value Pr(>|z|)
## (Intercept)       -1.569e+00  1.420e-01 -11.044  < 2e-16 ***
## make              -3.895e-01  2.315e-01  -1.683 0.092388 .
## address           -1.458e-01  6.928e-02  -2.104 0.035362 *
## all                1.141e-01  1.103e-01   1.035 0.300759
## num3d              2.252e+00  1.507e+00   1.494 0.135168
## our                5.624e-01  1.018e-01   5.524 3.31e-08 ***
## over               8.830e-01  2.498e-01   3.534 0.000409 ***
## remove             2.279e+00  3.328e-01   6.846 7.57e-12 ***
## internet           5.696e-01  1.682e-01   3.387 0.000707 ***
## order              7.343e-01  2.849e-01   2.577 0.009958 **
## mail               1.275e-01  7.262e-02   1.755 0.079230 .
## receive           -2.557e-01  2.979e-01  -0.858 0.390655
## will              -1.383e-01  7.405e-02  -1.868 0.061773 .
## people            -7.961e-02  2.303e-01  -0.346 0.729557
## report             1.447e-01  1.364e-01   1.061 0.288855
## addresses          1.236e+00  7.254e-01   1.704 0.088370 .
## free               1.039e+00  1.457e-01   7.128 1.01e-12 ***
## business           9.599e-01  2.251e-01   4.264 2.01e-05 ***
## email              1.203e-01  1.172e-01   1.027 0.304533
## you                8.131e-02  3.505e-02   2.320 0.020334 *
## credit             1.047e+00  5.383e-01   1.946 0.051675 .
## your               2.419e-01  5.243e-02   4.615 3.94e-06 ***
## font               2.013e-01  1.627e-01   1.238 0.215838
## num000             2.245e+00  4.714e-01   4.762 1.91e-06 ***
## money              4.264e-01  1.621e-01   2.630 0.008535 **
## hp                -1.920e+00  3.128e-01  -6.139 8.31e-10 ***
## hpl               -1.040e+00  4.396e-01  -2.366 0.017966 *
## george            -1.177e+01  2.113e+00  -5.569 2.57e-08 ***
## num650             4.454e-01  1.991e-01   2.237 0.025255 *
## lab               -2.486e+00  1.502e+00  -1.656 0.097744 .
## labs              -3.299e-01  3.137e-01  -1.052 0.292972
## telnet            -1.702e-01  4.815e-01  -0.353 0.723742
## num857             2.549e+00  3.283e+00   0.776 0.437566
## data              -7.383e-01  3.117e-01  -2.369 0.017842 *
## num415             6.679e-01  1.601e+00   0.417 0.676490
## num85             -2.055e+00  7.883e-01  -2.607 0.009124 **
## technology         9.237e-01  3.091e-01   2.989 0.002803 **
## num1999            4.651e-02  1.754e-01   0.265 0.790819
## parts             -5.968e-01  4.232e-01  -1.410 0.158473
## pm                -8.650e-01  3.828e-01  -2.260 0.023844 *
## direct            -3.046e-01  3.636e-01  -0.838 0.402215
## cs                -4.505e+01  2.660e+01  -1.694 0.090333 .
## meeting           -2.689e+00  8.384e-01  -3.207 0.001342 **
## original          -1.247e+00  8.064e-01  -1.547 0.121978
## project           -1.573e+00  5.292e-01  -2.973 0.002953 **
## re                -7.923e-01  1.556e-01  -5.091 3.56e-07 ***
## edu               -1.459e+00  2.686e-01  -5.434 5.52e-08 ***
## table             -2.326e+00  1.659e+00  -1.402 0.160958
## conference        -4.016e+00  1.611e+00  -2.493 0.012672 *
## charSemicolon     -1.291e+00  4.422e-01  -2.920 0.003503 **
## charRoundbracket  -1.881e-01  2.494e-01  -0.754 0.450663
## charSquarebracket -6.574e-01  8.383e-01  -0.784 0.432914
## charExclamation    3.472e-01  8.926e-02   3.890 0.000100 ***
```

```
## charDollar         5.336e+00  7.064e-01   7.553 4.24e-14 ***
## charHash           2.403e+00  1.113e+00   2.159 0.030883 *
## capitalAve         1.199e-02  1.884e-02   0.636 0.524509
## capitalLong        9.118e-03  2.521e-03   3.618 0.000297 ***
## capitalTotal       8.437e-04  2.251e-04   3.747 0.000179 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## (Dispersion parameter for binomial family taken to be 1)
##
##     Null deviance: 6170.2  on 4600  degrees of freedom
## Residual deviance: 1815.8  on 4543  degrees of freedom
## AIC: 1931.8
##
## Number of Fisher Scoring iterations: 13
```

The coefficient estimate in the output indicate the average change in the log odds of the response variable associated with a one unit increase in each predictor variable. For example, a one unit increase in the predictor variable chardollar is associated with an average change of 5.336e+00 in the log odds of the response variable am taking on a value of 1. This means that higher values of chardollar are associated with a lower likelihood of the type variable taking on a value of 1. The standard error gives us an idea of the variability associated with the coefficient estimate. We then divide the coefficient estimate by the standard error to obtain a z value. The p-value Pr(>|z|) tells us the probability associated with a particular z value. This essentially tells us how well each predictor variable is able to predict the value of the response variable in the model. For example, the p-value associated with the z value for the chardollar variable is 4.24e-14. Since this value is less than .0001, we would say that chardollar is a statistically significant predictor variable in the model. In the same way that we analyse the chardollar variable we can also find all the statistically significants predictors variables of the model in the summary to get a better understanding of our model.

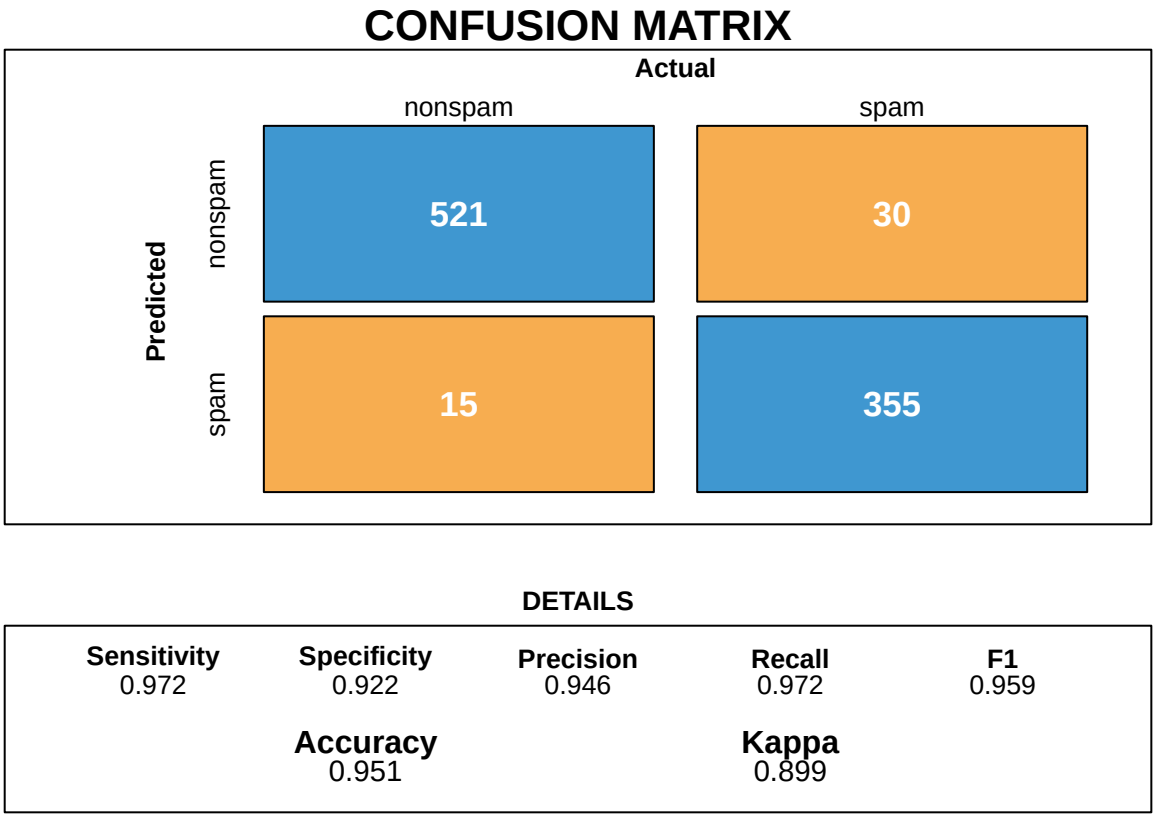
Let's build another model to check if we can outperform the model based on logistic regression. In this model we will use Random Forest :

```
#install.packages("randomForest")
library(randomForest)

spam_classifier <-  randomForest(factor(y_train$type) ~ ., data=X_train)


pred <- predict(spam_classifier, X_test)
cm <- confusionMatrix(data = as.factor(pred) , reference = as.factor(y_test$type))

draw_confusion_matrix(cm)
```




```
#install.packages("randomForest")
library(randomForest)

spam_classifier <-  randomForest(factor(y_train$type) ~ ., data=X_train)
```

We can see that the Random Forest model outperform the logistic regression model. The number of false-positive has decreased (from 41 to 30). The number of false-negative has decreased (from 23 to 15). We also improved the precision (from 92.6% to 94.4%) and the the recall (from 95.7% to 97.2%).

CONCLUSION:
We receive a dataset that we split into train and test set. We build a logistic regression model on the train set and verify the model performance on the test set . We tried and succeeded to improve the spam and non-spam classifier using a randomforest model.