Predictions Metrics: Classification

Oscar A. Trevizo

January 2023

Contents

Purpose

A function that assesses predictions by providing accuracy metrics.

There are two sets of metrics: One set for regression and one set for classification.

This notebook focuses on *classification metrics*.

Use this terminology:

- Prediction accuracy: Refers to regression
- Prediction metrics: Refers to classification

In this notebook, we fit a logistic regression model and evaluate performance of LDA, QDA and KNN classifiers.

There are various acronyms in the literature describing the same metrics. I am following the nomenclature from ISLR (see references below):

Formulas:

- OBS = Observations or truth cases
- Accuracy. ACC = sum(truth == predicted) * 100/length(truth)
- Sensitivity. TPR True Positive Rate = TP/(TP + FN) = TP/P
- Specificity. TNR True Negative Rate = TN/(FP + TN) = TN/N
- Precision. Positive Predictive Value. PPV = TP/(TP + FP)
- Negative Predictive Value. NPV = TN/(TN + FN)
- False Discovery Rate. FDR = FP/(TP + FP)
- False Positive Rate. FPR = FP/(FP + TN) = FP/N
- True Positives. TP = sum(truth == 1 & predicted == 1)
- True Negatives. TN = sum(truth == 0 & predicted == 0)
- False Positives. FP = sum(truth == 0 & predicted == 1)
- False Negatives. FN = sum(truth == 1 & predicted == 0)
- Positives. P = TP + FN # total number positives in the truth data
- Negatives. N = FP + TN # total number of negatives

Functions

Adapted from R functions shared by faculty in Harvard data science class (2021). See references at the bottom of this notebook.

```
###
#
# prediction.metrics function -- to return a list with all the metrics values
#
# Based on R functions shared by faculty in Harvard data science class (2021). See references.
#
# Input: truth and predicted lists.
#
# Returns a list with:
# [1] OBS = Observations or truth cases
# [2] Accuracy. ACC = sum(truth == predicted) * 100/length(truth)
# [3] Sensitivity. TPR True Positive Rate = TP/(TP + FN) = TP/P
# [4] Specificity. TNR True Negative Rate = TN/(FP + TN) = TN/N
# [5] Precision. Positive Predictive Value. PPV = TP/(TP + FP)
# [6] Negative Predictive Value. NPV = TN/(TN + FN)
# [7] False Discovery Rate. FDR = FP/(TP + FP)
# [8] False Positive Rate. FPR = FP/(FP + TN) = FP/N
# [9] True Positives. TP = sum(truth == 1 & predicted == 1)
# [10] True Negatives. TN = sum(truth == 0 & predicted == 0)
# [11] False Positives. FP = sum(truth == 0 & predicted == 1)
# [12] False Negatives. FN = sum(truth == 1 & predicted == 0)
# [13] Positives. P = TP + FN  # total number positives in the truth data
# [14] Negatives. N = FP + TN  # total number of negatives
#
prediction.metrics = function(truth, predicted) {
    # same length:
    if (length(truth) != length(predicted)) {
        stop("truth and predicted must be same length!")
    }
    # check for missing values (we are going to compute metrics on non-missing
    # values only)
    bKeep = !is.na(truth) & !is.na(predicted)
    predicted = predicted[bKeep]
    truth = truth[bKeep]
    # only 0 and 1:
    if (sum(truth %in% c(0, 1)) + sum(predicted %in% c(0, 1)) != 2 * length(truth)) {
        stop("only zeroes and ones are allowed!")
    }
    # how predictions align against known training/testing outcomes: TP/FP=
    # true/false positives, TN/FN=true/false negatives
    TP = sum(truth == 1 & predicted == 1)
    TN = sum(truth == 0 & predicted == 0)
    FP = sum(truth == 0 & predicted == 1)
    FN = sum(truth == 1 & predicted == 0)
    P = TP + FN  # total number of positives in the truth data
    N = FP + TN  # total number of negatives
    # Add the following output to return (OAT 11/9/2021)
    OBS = length(truth)
    ACC = sum(truth == predicted)/length(truth)
    TPR = TP/P
    TNR = TN/N
    PPV = TP/(TP + FP)
    NPV = TN/(TN + FN)
    FDR = FP/(TP + FP)
    FPR = FP/N

    # Returned a named list
    output <- list(OBS=OBS, ACC=ACC, TPR=TPR, TNR=TNR, PPV=PPV,
                   NPV=NPV, FDR=FDR, FPR=FPR, TP=TP,
                   TN=TN, FP=FP, FN=FN, P=P, N=N)
    return(output)
}


# Logistic regression
lgr.pred.ftn = function(formula, df.train, df.test){
  glm.fit <- glm(formula, data = df.train, family = binomial)
  glm.probs <- predict(glm.fit, newdata = df.test, type = "response")
  glm.pred <- rep(0, dim(df.test)[1])
  glm.pred[glm.probs>0.5]=1
  return(glm.pred)
}

# Linear Discriminant Analysis (LDA)
lda.pred.ftn = function(formula, df.train, df.test){
  lda.fit <- lda(formula, data = r3df.train)
  lda.pred <- predict(lda.fit, df.test)
  lda.class <- lda.pred$class
  return(lda.class)
}
```

```r
# Quadratic Discriminant Analysis (QDA)
qda.pred.ftn = function(formula, df.train, df.test){
  qda.fit <- qda(formula, data = r3df.train)
  qda.pred <- predict(qda.fit, df.test)
  qda.class <- qda.pred$class
  return(qda.class)
}
```

Simulate the data

Play with two sets of Normally distributed sets of data with different means. We can change the number of samples and we can move the means around.

```r
# From Harvard data science class (see references at the end of this notebook)

set.seed(11)

# Our measuring variable is continuous, numeric...
# ...it has two Normal distribution waves
x <- c(rnorm(30), rnorm(30, mean=2))

# Our outcome is categorical, A and B xxxx times each
# ...the idea is to match A and B to a number x
y <- rep(c("A", "B"), each=30)
```

Boxplot and histogram

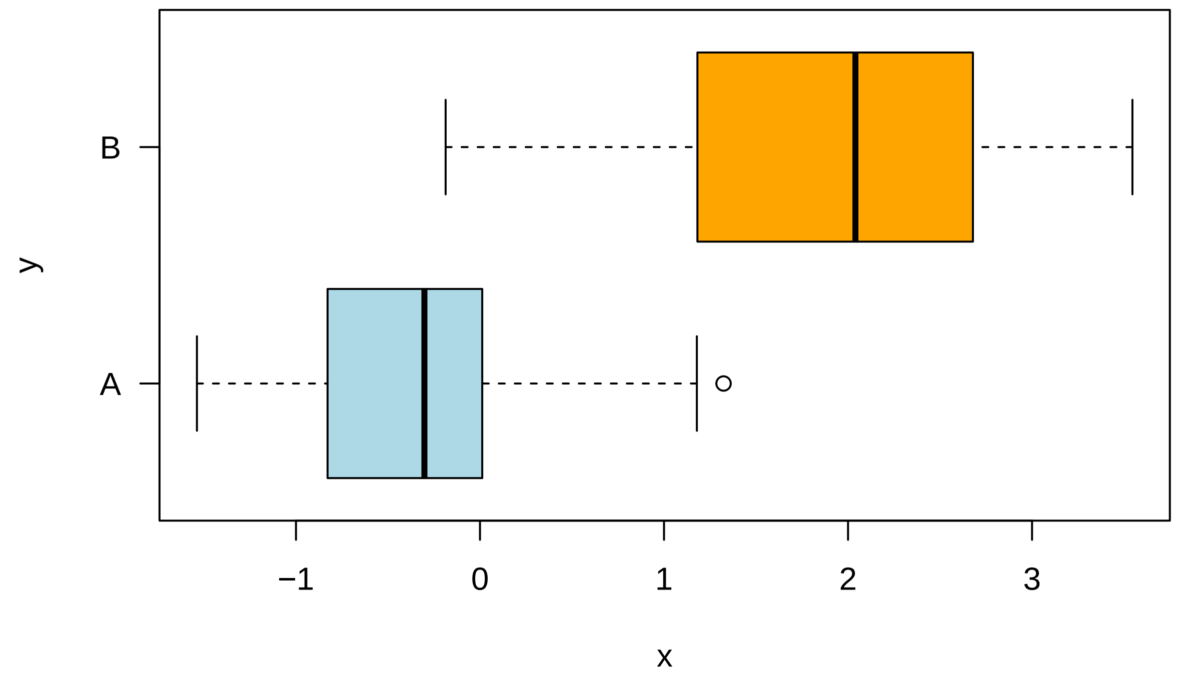
In a boxplot, we want to have the categorical variable in the horizontal axis.

That is why we see a formula x~y below.

The chart gives us more information if we plot it horizontally in this case.

The histogram adds information to visualize the behavior relationship between the outcome, categorical valuable, and predictor, numeric variable.

```r
# From Harvard data science class (see references)
boxplot(x~y, col=c("lightblue","orange"), horizontal=T, las=1)
```



```r
# Now place a histogram on top of another histogram

oldpar <- par(mfrow=c(3, 1), mar=c(2,2,1,1))

breaks <- seq(-2, 4, by=0.25)

# Histogram for 'B'
hist(x[y=="B"], breaks=breaks, col='orange', main="B", xaxt='n')

# Histogram for 'A'
hist(x[y=="A"], breaks=breaks, col='lightblue', main="A")

plot(x, ifelse(y=="A", 0,1), breaks=breaks, col=ifelse(y=="A", "lightblue","orange"), pch=19)
```

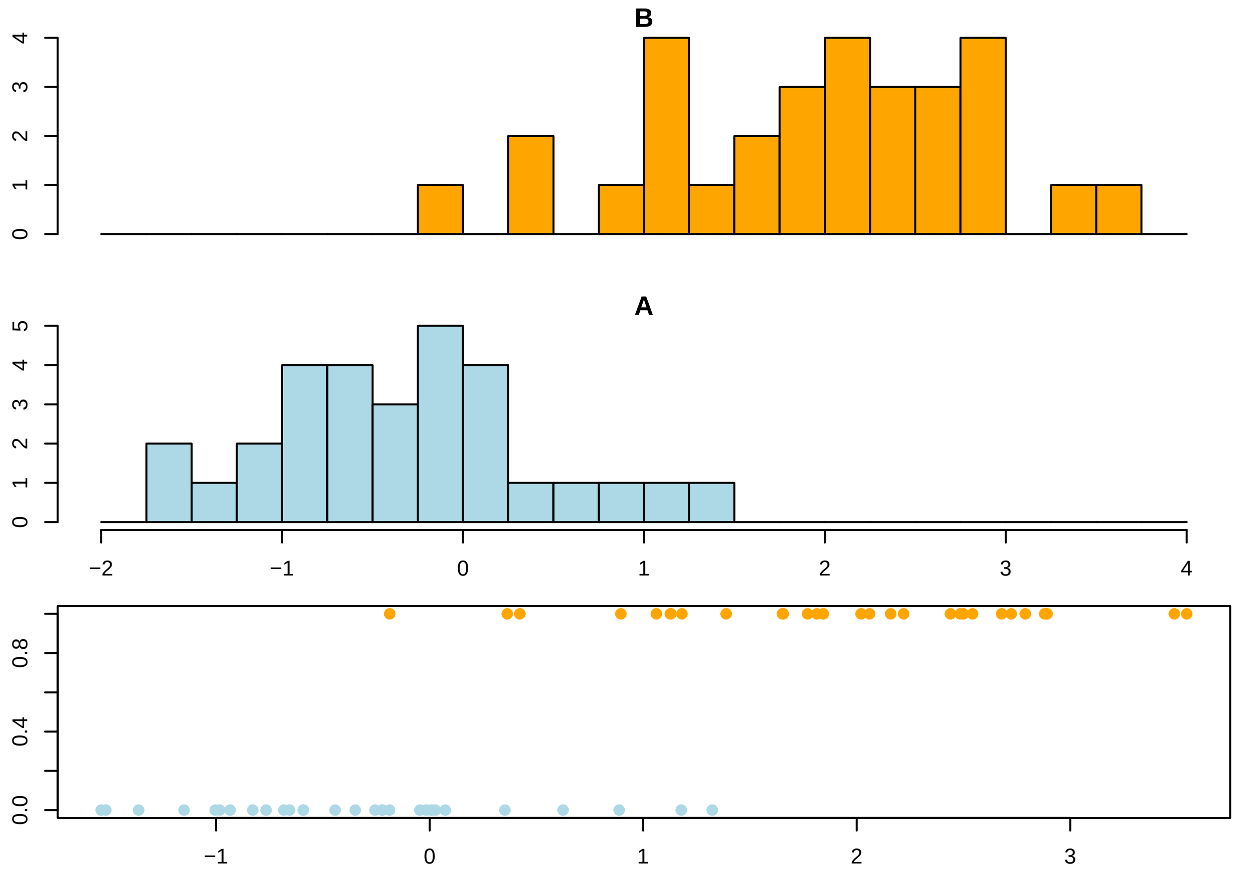
```
## Warning in plot.window(...): "breaks" is not a graphical parameter

## Warning in plot.xy(xy, type, ...): "breaks" is not a graphical parameter

## Warning in axis(side = side, at = at, labels = labels, ...): "breaks" is not a
## graphical parameter

## Warning in axis(side = side, at = at, labels = labels, ...): "breaks" is not a
## graphical parameter

## Warning in box(...): "breaks" is not a graphical parameter
## Warning in title(...): "breaks" is not a graphical parameter
```



```
par(oldpar)
```

Linear Discriminant Analysis (LDA)

Needs library{MASS}

```
##
#
# LDA from library{MASS}
#
##
lda.m <- lda(y~x, data=data.frame(y=y, x=x))
lda.m
```

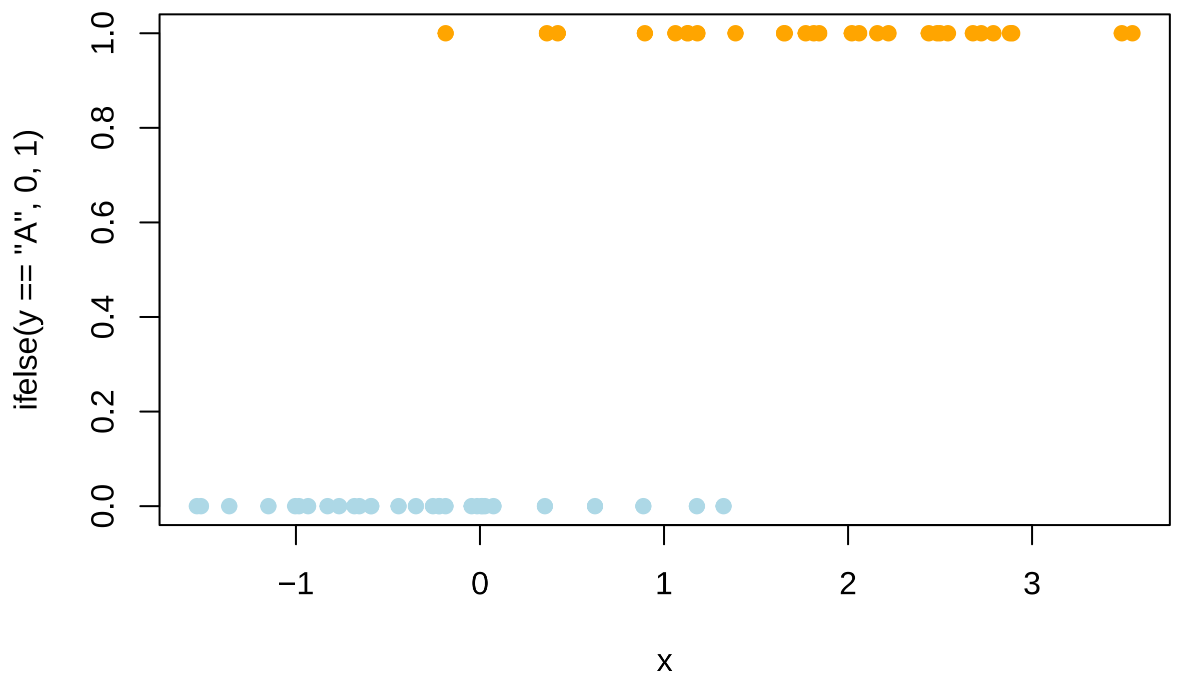
```
## Call:
## lda(y ~ x, data = data.frame(y = y, x = x))
##
## Prior probabilities of groups:
##   A   B
## 0.5 0.5
##
```

```
## Group means:
##            x
## A -0.3285999
## B  1.9477643
##
## Coefficients of linear discriminants:
##        LD1
## x 1.20728
```

```
# summary(m)
```

Plot results

```
plot(x, ifelse(y=="A", 0,1), col=ifelse(y=="A", "lightblue","orange"), pch=19)
```



```
# theoretical probability of B, TRUTH (dashed line):
# points(xx,exp(-(xx-2)^2/2)*0.75/(exp(-(xx-2)^2/2)*0.75+exp(-xx^2/2)*0.25),type='l',col='red',lwd=2,lt

# from estimated distributions, LDA-style (from slide 16):
# points(xx,exp(-(xx-m.b)^2/(2*s^2))*p.b/
# (exp(-(xx-m.b)^2/(2*s^2))*p.b+
# exp(-(xx-m.a)^2/(2*s^2))*p.a),
# type='l',col='red',lwd=2)
```

Predict

```r
lda.p <- predict(lda.m)
```

Confusion matrix

```r
lda.class = lda.p$class
table(lda.class, y)
```

```
##          y
## lda.class  A  B
##         A 27  3
##         B  3 27
```

```r
mean(lda.class == y)
```

```
## [1] 0.9
```

Prediction metrics

```r
lda.metrics <- prediction.metrics(ifelse(y=="A",0,1), ifelse(lda.class=="A",0,1))
lda.metrics
```

```
## $OBS
## [1] 60
##
## $ACC
## [1] 0.9
##
## $TPR
## [1] 0.9
##
## $TNR
## [1] 0.9
##
## $PPV
## [1] 0.9
##
## $NPV
## [1] 0.9
##
## $FDR
## [1] 0.1
##
## $FPR
## [1] 0.1
```

```
##
## $TP
## [1] 27
##
## $TN
## [1] 27
##
## $FP
## [1] 3
##
## $FN
## [1] 3
##
## $P
## [1] 30
##
## $N
## [1] 30
```

References

- Harvard “Elements of Statistical Learning” (2021) taught by professors Dr. Sivachenko, Dr. Farutin
- Book “An Introduction to Statistical Learning with Applications in R” (ISLR) by Gareth James et al