Data 607 Week 2B Classification Metrics

Benjamin Bravo

2026-02-05

Approach Deliverable

Approach

This assignment evaluates a binary classification model and examines how probability thresholds affect performance. I will begin by exploring the dataset, visualizing the actual class distribution, and calculating the null error rate to establish a baseline.

Next, I will use the predicted probabilities to generate class predictions at thresholds of 0.2, 0.5, and 0.8. For each threshold, I will compute confusion matrices and calculate accuracy, precision, recall, and F1 score. The results will be summarized in a table and interpreted using real-world examples to explain threshold tradeoffs.

Anticipated Data Challenges

One challenge is class imbalance, which can make accuracy misleading and requires careful interpretation of metrics like precision and recall. Another challenge is choosing appropriate thresholds, since changing the threshold can improve one metric while worsening another. Addressing these issues requires focusing on the confusion matrix and understanding the context of model use.

Introduction

This dataset includes model predictions and true labels used to evaluate a binary classification task. It contains the predicted probability of an observation being female, the corresponding predicted class based on a default threshold, and the actual sex label used during model training. ## Link: https://raw.githubusercontent.com/bb2955/607-Week2/main/penguin_predictions.csv

```
install.packages("tidyverse")
```

```
## Installing package into '/cloud/lib/x86_64-pc-linux-gnu-library/4.5'
## (as 'lib' is unspecified)
```

```
library(tidyverse)
```

```
## -- Attaching core tidyverse packages ------------------------ tidyverse 2.0.0 --
## v dplyr     1.1.4     v readr     2.1.6
## v forcats   1.0.1     v stringr   1.6.0
## v ggplot2   4.0.1     v tibble    3.3.0
## v lubridate 1.9.4     v tidyr     1.3.2
## v purrr     1.2.0
## -- Conflicts ------------------------------------------ tidyverse_conflicts() --
## x dplyr::filter() masks stats::filter()
## x dplyr::lag()    masks stats::lag()
## i Use the conflicted package (<http://conflicted.r-lib.org/>) to force all conflicts to become errors
```

```r
data_url <- "https://raw.githubusercontent.com/bb2955/607-Week2/main/penguin_predictions.csv"

df <- readr::read_csv(data_url, show_col_types = FALSE)
```

```r
install.packages("conflicted")
```

```
## Installing package into '/cloud/lib/x86_64-pc-linux-gnu-library/4.5'
## (as 'lib' is unspecified)
```

```r
library(conflicted)

conflict_prefer_all("dplyr", quiet = TRUE)
```

```r
install.packages("ggplot2")
```

```
## Installing package into '/cloud/lib/x86_64-pc-linux-gnu-library/4.5'
## (as 'lib' is unspecified)
```

```r
library(ggplot2)
```

```r
str(df)
```

```
## spc_tbl_ [93 x 3] (S3: spec_tbl_df/tbl_df/tbl/data.frame)
##  $ .pred_female: num [1:93] 0.992 0.954 0.985 0.187 0.995 ...
##  $ .pred_class : chr [1:93] "female" "female" "female" "male" ...
##  $ sex         : chr [1:93] "female" "female" "female" "female" ...
##  - attr(*, "spec")=
##   .. cols(
##   ..   .pred_female = col_double(),
##   ..   .pred_class = col_character(),
##   ..   sex = col_character()
##   .. )
##  - attr(*, "problems")=<externalptr>
```

```r
table(df$sex)
```

```
##
## female   male
##     39     54
```

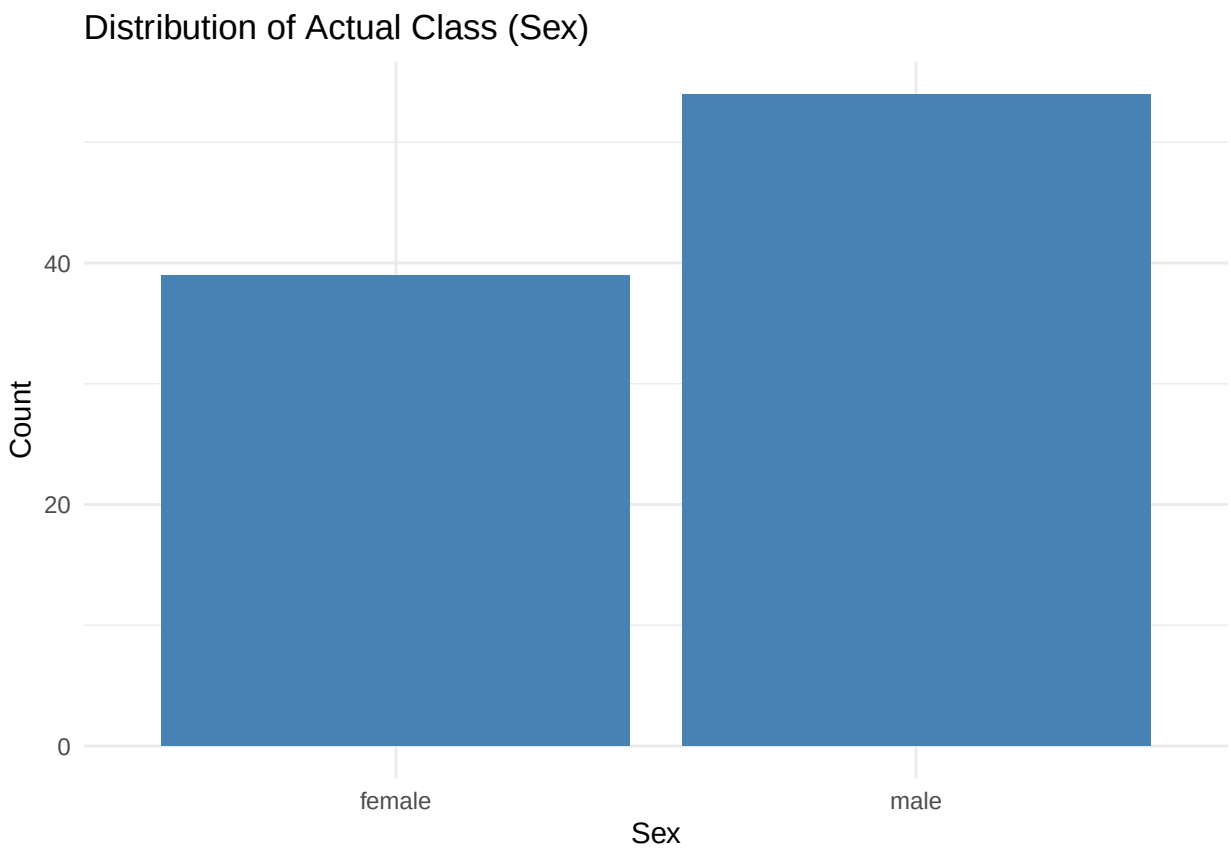
Null Error Rate

```r
class_props <- prop.table(table(df$sex))
```

```r
null_error_rate <- 1 - max(class_props)
null_error_rate
```

```
## [1] 0.4193548
```

```r
ggplot(df, aes(x = factor(sex))) +
  geom_bar(fill = "steelblue") +
  labs(
    title = "Distribution of Actual Class (Sex)",
    x = "Sex",
    y = "Count"
  ) +
  theme_minimal()
```



Confusion Matrices at Multiple Thresholds

Recode sex into numeric otherwise sex is not coded as 0/1 in the dataset

```
df <- df %>%
  mutate(
    sex_num = ifelse(sex == "female", 1, 0)
  )
```

```
confusion_counts <- function(threshold) {

  pred <- ifelse(df$.pred_female > threshold, 1, 0)

  TP <- sum(pred == 1 & df$sex_num == 1)
  FP <- sum(pred == 1 & df$sex_num == 0)
  TN <- sum(pred == 0 & df$sex_num == 0)
  FN <- sum(pred == 0 & df$sex_num == 1)

  c(TP = TP, FP = FP, TN = TN, FN = FN)
}
```

```
cm_02 <- confusion_counts(0.2)
cm_05 <- confusion_counts(0.5)
cm_08 <- confusion_counts(0.8)

cm_02
```

```
## TP FP TN FN
```

```
## 37  6 48  2
```

```
cm_05
```

```
## TP FP TN FN
## 36  3 51  3
```

```
cm_08
```

```
## TP FP TN FN
## 36  2 52  3
```

Performance Metrics

```
metrics <- function(cm) {
  TP <- cm["TP"]
  FP <- cm["FP"]
  TN <- cm["TN"]
  FN <- cm["FN"]

  accuracy  <- (TP + TN) / sum(cm)
  precision <- TP / (TP + FP)
  recall    <- TP / (TP + FN)
  f1        <- 2 * precision * recall / (precision + recall)

  c(
    Accuracy = accuracy,
    Precision = precision,
    Recall = recall,
    F1 = f1
  )
}
```

```
metrics_02 <- metrics(cm_02)
metrics_05 <- metrics(cm_05)
metrics_08 <- metrics(cm_08)
```

```
results <- rbind(
  Threshold_0.2 = metrics_02,
  Threshold_0.5 = metrics_05,
  Threshold_0.8 = metrics_08
)

results
```

```
##               Accuracy.TP Precision.TP Recall.TP     F1.TP
## Threshold_0.2   0.9139785    0.8604651 0.9487179 0.9024390
## Threshold_0.5   0.9354839    0.9230769 0.9230769 0.9230769
## Threshold_0.8   0.9462366    0.9473684 0.9230769 0.9350649
```

Threshold Use Cases

When a 0.2 Threshold is Preferable

A low threshold is useful when it is more important to catch as many positive cases as possible, even if some mistakes are made. For example, in a medical screening test, doctors would rather flag someone who might be sick and run more tests than miss someone who actually has the disease.

When a 0.8 Threshold Is Preferable

A high threshold is useful when you only want to make a positive prediction when you are very confident. For example, in a spam filter or fraud detection system, it is better to avoid incorrectly flagging legitimate emails or transactions, even if that means missing some true spam or fraud.

Conclusion

The results show that model performance changes as the probability threshold increases. Accuracy improves slightly as the threshold moves from 0.2 to 0.8, while precision increases and recall slightly decreases. This means the model becomes more confident in its positive predictions at higher thresholds but may miss some positive cases.

These findings suggest that the threshold should be chosen based on the goal of the analysis. If identifying as many positive cases as possible is important, a lower threshold is more appropriate. If making fewer but more reliable positive predictions is the priority, a higher threshold works better. This analysis could be extended by testing more thresholds, using ROC or precision-recall curves, or validating the model on new data.