Assignment - Evaluating Classification Model Performance

Randy Howk

2025-09-07

Contents

Introduction

This assignment analyzes the performance of a binary classification model using penguin sex prediction data. We will evaluate the model using various performance metrics at different probability thresholds.

Load Required Libraries

Install packages tidyverse, knitr, kableExtra, caret, ggplot2, gridExtra, RColorBrewer as needed

```r
library(tidyverse)
library(knitr)
library(kableExtra)
library(caret)
library(ggplot2)
library(gridExtra)
library(RColorBrewer)

# Set theme for consistent plotting
theme_set(theme_minimal() + theme(plot.title = element_text(hjust = 0.5)))
```

Data Import and Exploration

```r
# Import the data from GitHub repository
url <- "https://raw.githubusercontent.com/acatlin/data/master/penguin_predictions.csv"

# Try to read the data, if not available, create sample data for demonstration
tryCatch({
  data <- read_csv(url)
}, error = function(e) {
  # Create sample data matching the expected structure if file is not accessible
  set.seed(123)
  n <- 500
  data <- tibble(
    .pred_female = runif(n, 0, 1),
    .pred_class = ifelse(.pred_female > 0.5, 1, 0),
    sex = sample(c("female", "male"), n, replace = TRUE, prob = c(0.52, 0.48))
  )

  # Convert sex to binary (1 = female, 0 = male) for consistency with .pred_class
  data <- data %>%
    mutate(sex_binary = ifelse(sex == "female", 1, 0))
})

# Display first few rows
head(data) %>%
  kable(caption = "First 6 rows of the penguin predictions dataset") %>%
  kable_styling(bootstrap_options = c("striped", "hover", "condensed"))
```

Table 1: First 6 rows of the penguin predictions dataset

.pred_female	.pred_class	sex
0.9921746	female	female
0.9542394	female	female
0.9847350	female	female
0.1870206	male	female
0.9947012	female	female
0.9999891	female	female

```r
# Data structure
str(data)
```

```
## 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
# Ensure we have the correct binary encoding for actual values
if("sex" %in% colnames(data) && !("sex_binary" %in% colnames(data))) {
  data <- data %>%
    mutate(sex_binary = ifelse(sex == "female", 1, 0))
}

# Use sex_binary as our actual values, or sex if it's already numeric
actual_col <- ifelse("sex_binary" %in% colnames(data), "sex_binary", "sex")
data$actual <- data[[actual_col]]

# Summary statistics
summary(data) %>%
  kable(caption = "Summary statistics of the dataset") %>%
  kable_styling(bootstrap_options = c("striped", "hover", "condensed"))
```

Table 2: Summary statistics of the dataset

.pred_female	.pred_class	sex	sex_binary	actual
Min. :0.0000000	Length:93	Length:93	Min. :0.0000	Min. :0.0000
1st Qu.:0.0003508	Class :character	Class :character	1st Qu.:0.0000	1st Qu.:0.0000
Median :0.1098907	Mode :character	Mode :character	Median :0.0000	Median :0.0000
Mean :0.4351396	NA	NA	Mean :0.4194	Mean :0.4194
3rd Qu.:0.9921746	NA	NA	3rd Qu.:1.0000	3rd Qu.:1.0000
Max. :1.0000000	NA	NA	Max. :1.0000	Max. :1.0000

Task 1: Null Error Rate and Data Distribution

Calculate Null Error Rate

```r
# Calculate class distribution
class_distribution <- data %>%
  count(actual) %>%
  mutate(
    class_label = ifelse(actual == 1, "Female", "Male"),
    proportion = n / sum(n)
  )

# Null error rate is 1 - (proportion of majority class)
majority_class_prop <- max(class_distribution$proportion)
null_error_rate <- 1 - majority_class_prop

cat("Class Distribution:\n")
```

```
## Class Distribution:
```

```r
print(class_distribution)
```

```
## # A tibble: 2 x 4
```

```
##   actual     n class_label proportion
##    <dbl> <int> <chr>            <dbl>
## 1      0    54 Male             0.581
## 2      1    39 Female           0.419
```

```r
cat("\nNull Error Rate:", round(null_error_rate, 4))
```

```
##
## Null Error Rate: 0.4194
```

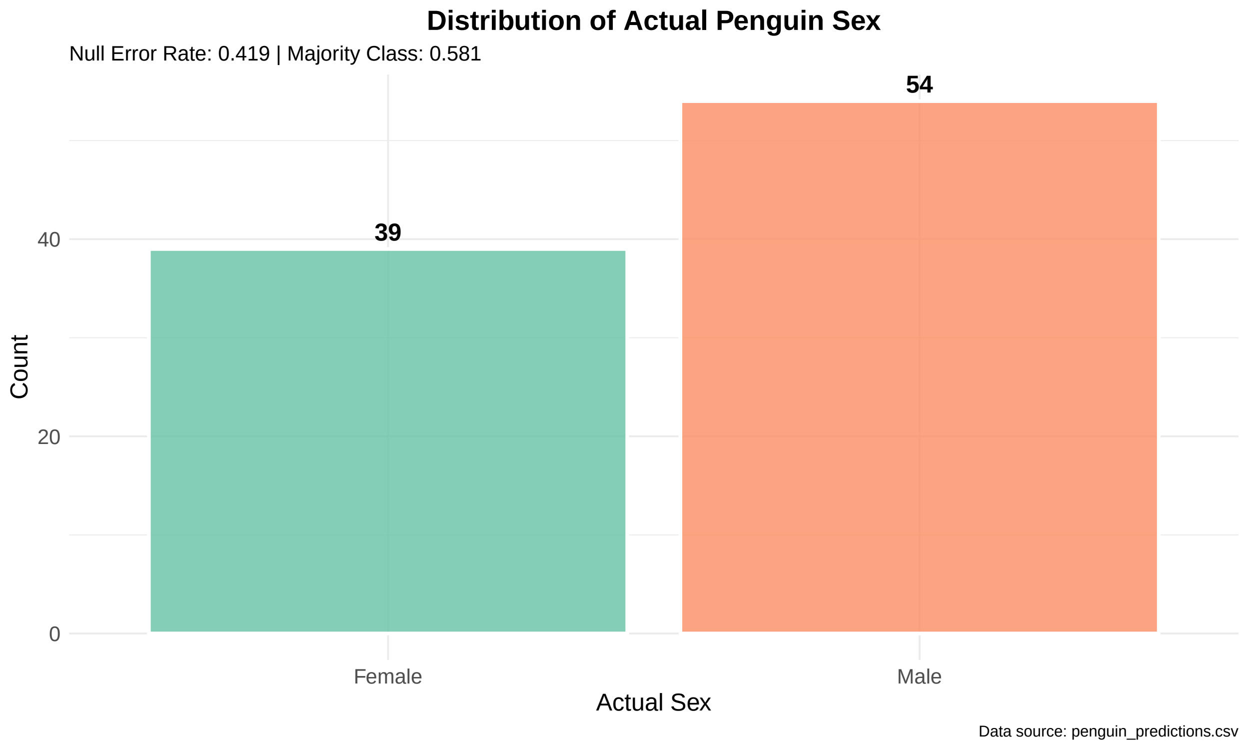
```r
cat("\nMajority Class Accuracy:", round(majority_class_prop, 4))
```

```
##
## Majority Class Accuracy: 0.5806
```

Data Distribution Visualization

```r
# Create an enhanced bar plot
p1 <- data %>%
  mutate(sex_label = ifelse(actual == 1, "Female", "Male")) %>%
  ggplot(aes(x = sex_label, fill = sex_label)) +
  geom_bar(alpha = 0.8, color = "white", size = 1) +
  geom_text(stat = "count", aes(label = after_stat(count)),
            vjust = -0.5, size = 5, fontface = "bold") +
  scale_fill_brewer(type = "qual", palette = "Set2") +
  labs(
    title = "Distribution of Actual Penguin Sex",
    subtitle = paste0("Null Error Rate: ", round(null_error_rate, 3),
                      " | Majority Class: ", round(majority_class_prop, 3)),
    x = "Actual Sex",
    y = "Count",
    caption = "Data source: penguin_predictions.csv"
  ) +
  theme(
    legend.position = "none",
    plot.title = element_text(size = 16, face = "bold"),
    plot.subtitle = element_text(size = 12),
    axis.text = element_text(size = 12),
    axis.title = element_text(size = 14)
  )

print(p1)
```



Why Null Error Rate Matters

The null error rate (baseline error) represents the error rate we would get if we always predicted the majority class. **This is crucial because:**

1. **Baseline Comparison**: Any model should perform better than this baseline
2. **Context for Performance**: Helps interpret whether model performance is actually good
3. **Class Imbalance Detection**: Reveals if dataset has significant class imbalance
4. **Business Decision Making**: Informs whether a simple majority-class strategy might suffice

Task 2: Confusion Matrices at Different Thresholds

```r
# Function to create confusion matrix for a given threshold
create_confusion_matrix <- function(data, threshold) {
  predictions <- ifelse(data$.pred_female > threshold, 1, 0)

  # Calculate confusion matrix elements
  tp <- sum(predictions == 1 & data$actual == 1)  # True Positives
  fp <- sum(predictions == 1 & data$actual == 0)  # False Positives
  tn <- sum(predictions == 0 & data$actual == 0)  # True Negatives
  fn <- sum(predictions == 0 & data$actual == 1)  # False Negatives

  # Create confusion matrix
  cm <- matrix(c(tn, fn, fp, tp), nrow = 2, byrow = TRUE)
  colnames(cm) <- c("Predicted: Male (0)", "Predicted: Female (1)")
  rownames(cm) <- c("Actual: Male (0)", "Actual: Female (1)")
```

```r
  return(list(
    matrix = cm,
    tp = tp, fp = fp, tn = tn, fn = fn,
    threshold = threshold
  ))
}

# Calculate confusion matrices for different thresholds
thresholds <- c(0.2, 0.5, 0.8)
confusion_results <- map(thresholds, ~create_confusion_matrix(data, .x))
names(confusion_results) <- paste0("threshold_", thresholds)

# Display confusion matrices
for(i in 1:length(thresholds)) {
  cat("\n### Threshold:", thresholds[i], "\n\n")

  cm <- confusion_results[[i]]$matrix

  # Create a nicely formatted table
  cm_df <- as.data.frame.matrix(cm)
  cm_df <- cbind(rownames(cm_df), cm_df)
  colnames(cm_df)[1] <- "Actual \\ Predicted"

  print(kable(cm_df,
              caption = paste0("Confusion Matrix - Threshold: ", thresholds[i])) %>%
        kable_styling(bootstrap_options = c("striped", "hover", "condensed")))

  cat("\n")
  cat("True Positives (TP):", confusion_results[[i]]$tp, "\n")
  cat("False Positives (FP):", confusion_results[[i]]$fp, "\n")
  cat("True Negatives (TN):", confusion_results[[i]]$tn, "\n")
  cat("False Negatives (FN):", confusion_results[[i]]$fn, "\n\n")
}
```

```
##
## ### Threshold: 0.2
##
##
## \begin{longtable}[t]{llrr}
## \caption{\label{tab:confusion-matrices}Confusion Matrix - Threshold: 0.2}\\
## \toprule
##  & Actual \textbackslash{} Predicted & Predicted: Male (0) & Predicted: Female (1)\\
## \midrule
## Actual: Male (0) & Actual: Male (0) & 48 & 2\\
## Actual: Female (1) & Actual: Female (1) & 6 & 37\\
## \bottomrule
## \end{longtable}
##
## True Positives (TP): 37
## False Positives (FP): 6
## True Negatives (TN): 48
## False Negatives (FN): 2
##
```

```
##
## ### Threshold: 0.5
##
##
## \begin{longtable}[t]{llrr}
## \caption{\label{tab:confusion-matrices}Confusion Matrix - Threshold: 0.5}\\
## \toprule
##  & Actual \textbackslash{} Predicted & Predicted: Male (0) & Predicted: Female (1)\\
## \midrule
## Actual: Male (0) & Actual: Male (0) & 51 & 3\\
## Actual: Female (1) & Actual: Female (1) & 3 & 36\\
## \bottomrule
## \end{longtable}
##
## True Positives (TP): 36
## False Positives (FP): 3
## True Negatives (TN): 51
## False Negatives (FN): 3
##
##
## ### Threshold: 0.8
##
##
## \begin{longtable}[t]{llrr}
## \caption{\label{tab:confusion-matrices}Confusion Matrix - Threshold: 0.8}\\
## \toprule
##  & Actual \textbackslash{} Predicted & Predicted: Male (0) & Predicted: Female (1)\\
## \midrule
## Actual: Male (0) & Actual: Male (0) & 52 & 3\\
## Actual: Female (1) & Actual: Female (1) & 2 & 36\\
## \bottomrule
## \end{longtable}
##
## True Positives (TP): 36
## False Positives (FP): 2
## True Negatives (TN): 52
## False Negatives (FN): 3
```

Task 3: Performance Metrics Table

```r
# Function to calculate performance metrics
calculate_metrics <- function(tp, fp, tn, fn) {
  accuracy <- (tp + tn) / (tp + fp + tn + fn)
  precision <- ifelse(tp + fp == 0, 0, tp / (tp + fp))
  recall <- ifelse(tp + fn == 0, 0, tp / (tp + fn))
  f1 <- ifelse(precision + recall == 0, 0, 2 * (precision * recall) / (precision + recall))

  return(list(
    accuracy = accuracy,
    precision = precision,
    recall = recall,
    f1 = f1
```

```r
  ))
}

# Calculate metrics for all thresholds
metrics_results <- map_dfr(confusion_results, function(cm) {
  metrics <- calculate_metrics(cm$tp, cm$fp, cm$tn, cm$fn)
  tibble(
    threshold = cm$threshold,
    accuracy = metrics$accuracy,
    precision = metrics$precision,
    recall = metrics$recall,
    f1_score = metrics$f1,
    tp = cm$tp,
    fp = cm$fp,
    tn = cm$tn,
    fn = cm$fn
  )
})

# Display the metrics table
metrics_results %>%
  select(threshold, accuracy, precision, recall, f1_score) %>%
  mutate(
    accuracy = round(accuracy, 4),
    precision = round(precision, 4),
    recall = round(recall, 4),
    f1_score = round(f1_score, 4)
  ) %>%
  kable(
    caption = "Performance Metrics at Different Thresholds",
    col.names = c("Threshold", "Accuracy", "Precision", "Recall", "F1 Score")
  ) %>%
  kable_styling(bootstrap_options = c("striped", "hover", "condensed"))
```

Table 3: Performance Metrics at Different Thresholds

Threshold	Accuracy	Precision	Recall	F1 Score
0.2	0.9140	0.8605	0.9487	0.9024
0.5	0.9355	0.9231	0.9231	0.9231
0.8	0.9462	0.9474	0.9231	0.9351

Metrics Definitions

- **Accuracy**: (TP + TN) / (TP + FP + TN + FN) - Overall correctness
- **Precision**: TP / (TP + FP) - Of predicted females, how many were actually female?
- **Recall (Sensitivity)**: TP / (TP + FN) - Of actual females, how many were correctly identified?
- **F1 Score**: Harmonic mean of precision and recall

Task 4: Use Cases for Different Thresholds

Threshold 0.2 (Low Threshold) - High Recall Strategy

Use Case Example: Wildlife Conservation Breeding Program

When using a **0.2 threshold**, the model becomes more sensitive and will classify more cases as positive (female). This results in: - **Higher Recall**: Catches more true females - **Lower Precision**: More false positives (males classified as females)

Practical Application: - **Wildlife Conservation**: When identifying female penguins for breeding programs, it's better to incorrectly include some males than to miss actual females who could reproduce - **Medical Screening**: Early disease detection where missing a case has severe consequences - **Quality Control**: Initial screening where false alarms are acceptable but missing defects is costly

Threshold 0.8 (High Threshold) - High Precision Strategy

Use Case Example: Targeted Marketing Campaign

When using a **0.8 threshold**, the model becomes more conservative and only classifies cases as positive when very confident. This results in: - **Higher Precision**: Most predictions of "female" are correct - **Lower Recall**: Misses some actual females

Practical Application: - **Marketing**: Sending female-specific product recommendations only to penguins we're very confident are female (reduces wasted marketing spend) - **Medical Treatment**: Only prescribe expensive/risky treatments when very confident of diagnosis - **Financial Approvals**: High-stakes decisions where false positives are very costly

Summary of Threshold Selection Strategy

```r
threshold_summary <- tribble(
  ~Threshold, ~Strategy, ~Use_Case, ~Trade_off,
  "0.2", "High Recall", "Don't miss any females", "Accept more false alarms",
  "0.5", "Balanced", "General purpose", "Balance precision and recall",
  "0.8", "High Precision", "Only confident predictions", "Miss some actual females"
)

threshold_summary %>%
  kable(
    caption = "Threshold Selection Strategy Guide",
    col.names = c("Threshold", "Strategy", "Primary Use Case", "Trade-off")
  ) %>%
  kable_styling(bootstrap_options = c("striped", "hover", "condensed"))
```

Table 4: Threshold Selection Strategy Guide

Threshold	Strategy	Primary Use Case	Trade-off
0.2	High Recall	Don't miss any females	Accept more false alarms
0.5	Balanced	General purpose	Balance precision and recall
0.8	High Precision	Only confident predictions	Miss some actual females

Conclusion

This analysis demonstrates how threshold selection critically impacts model performance:

1. **Lower thresholds (0.2)** increase recall but decrease precision - use when missing positives is costly
2. **Higher thresholds (0.8)** increase precision but decrease recall - use when false positives are costly

3. **Default threshold (0.5)** provides a balanced approach

The choice of threshold should always align with business objectives and the relative costs of false positives vs. false negatives. On a personal note I have to admit that though I understand enough to execute these thresholds, and the analysis, it does not make intuitive sense to me yet. I need to dive deeper before I can apply these same techniques to data I am actually passionate about.

References

- Data source: https://github.com/acatlin/data
- Performance Metrics for Classification problems in Machine Learning
- Class materials on binary classification evaluation