Performance Metrics f...
PDF

week 2b_ Evaluating Cl...
PDF

remeber the content

Updated saved memory

Thought for 11s >

Done — I've saved the key content from both PDFs:

- **Classification metrics PDF**: confusion matrix (TP/FP/TN/FN), Accuracy/Precision/Recall(Sensitivity)/Specificity/F1, and the "minimize FN vs minimize FP depending on context" idea. Performance Metrics for Classif...
- **Week 2b assignment PDF**: dataset columns (`.pred_female` , `.pred_class` , `sex`), tasks (null error rate + `sex` plot, confusion matrices at thresholds 0.2/0.5/0.8, metrics table, real-world threshold scenarios), and the recommended step-by-step workflow. week 2b_ Evaluating Classificat...

Sources

What should i do with the NER

Thought for 11s >

NER = **Null Error Rate** — your **baseline error** if you ignore the model and **always predict the majority class**.

What you should do with the NER (in your Week 2b write-up)

1) Compute it (using *only* the true labels)

- Count how many **male** and **female** you have in `sex` (after handling missing `sex` if any).
- Let:
  - `N = total observations`
  - `majority = max(#male, #female)`
- Then:
  - **Null Accuracy** = `majority / N`
  - **NER (Null Error Rate)** = `1 - majority / N` (equivalently, `minority / N`)

Using your counts: `male = 54`, `female = 39`, so `N = 93`

- Majority = 54 (male)
- Null Accuracy = 54 / 93 = 0.580645… ≈ **0.581**
- **NER** = 1 − 0.580645… = 0.419354… ≈ **0.419**

Interpretation: if you naïvely predict **"male" for everyone**, you'll still be wrong about **41.9%** of the time.

2) Report it as your "do-nothing" benchmark

NER answers: **"How bad is it if I do the simplest possible thing?"**
This is important because model metrics (especially Accuracy) can look "good" just because one class is more common.

In your report, you usually include 1–2 sentences like:

- "The null model that predicts the majority class achieves ~0.581 accuracy (NER ~0.419). Any useful classifier should beat this baseline."

3) Use it to evaluate your model later (after you compute metrics at thresholds)

When you compute Accuracy at thresholds **0.2 / 0.5 / 0.8**, compare each Accuracy to the **Null Accuracy**:

- If **Accuracy ≤ Null Accuracy (≈ 0.581)** → the model (at that threshold) is **not better than guessing the majority class**.

- If **Accuracy is only slightly above 0.581** → the model may be only marginally useful; you should look closely at **Precision/Recall/F1**.
- Even if Accuracy is not much higher, the model can still be valuable if it improves what you care about (e.g., **Recall** for the positive class).

So NER is not “extra”—it’s the **baseline you compare against**.

Is NER the same step as the sex bar plot?

They’re typically presented together, but they’re **not the same thing**:

- **Sex distribution plot**: visual evidence of class balance/imbalance.
- **NER**: one-number summary baseline derived from that distribution.

You generally include **both**:

- Plot → shows imbalance clearly.
- NER → gives the baseline number you’ll use later.

Practical R template (you can paste and adapt)

```r
library(dplyr)

# assuming your data frame is penguin_predictions
df <- penguin_predictions %>%
  filter(!is.na(sex))  # if needed

sex_counts <- df %>% count(sex)

N <- sum(sex_counts$n)
majority <- max(sex_counts$n)

null_accuracy <- majority / N
NER <- 1 - null_accuracy

sex_counts
```

```
null_accuracy
NER
```

Copy code

If your plot already told you `male=54, female=39`, then your computed NER should match **~0.419**.

What to write (simple, acceptable wording)

- "The dataset has 54 male and 39 female observations (N=93). The null classifier that always predicts the majority class (male) achieves 0.581 accuracy, corresponding to a null error rate (NER) of 0.419. This baseline is used to judge whether our threshold-based classifiers meaningfully improve performance."

If you want, paste your first few rows (`head()`) or your `sex_counts` output, and I'll sanity-check that your NER matches the dataset after NA handling._

4/4

xplain why knowing the null error rate is important when evaluating models.
I dont understand

Null Error Rate (NER) is important because it tells you **how well you can do without a model**. It's the **baseline** you must beat.

1) Without NER, you can be fooled by "good accuracy"

If one class is more common, you can get decent accuracy by **always guessing the majority class**.

Example (your data idea):

- male = 54, female = 39 (total 93)
- If you always predict **male**:
  - Accuracy = 54/93 ≈ 0.58

- NER = 1 − 0.58 ≈ 0.42

So **0.58 accuracy is not impressive**, because you got it with **zero intelligence**.

Key point: A model with 0.60 accuracy might *sound* good, but it's only **barely better than the "always male" dummy rule**.

2) NER answers: "Is my model doing anything useful at all?"

When you evaluate your real model:

- If your model's accuracy is **below or equal to** the null accuracy (1 − NER),
  → your model is **not better than guessing the majority class**.

So NER is a quick check to avoid wasting time on a model that doesn't beat a trivial baseline.

3) It tells you when Accuracy is the wrong metric to focus on

When classes are imbalanced, accuracy can hide failures.

Example:

- Suppose a disease happens in 1% of people.
- Always predict "no disease":
  - Accuracy = 99%
  - NER = 1%
    Sounds amazing, but it detects **0** sick people (Recall = 0).

NER helps you realize:

- "High accuracy might just be majority guessing."
- So you should look at **Precision/Recall/F1**, not just accuracy.

4) It helps you explain threshold choices (0.2 / 0.5 / 0.8)

In your assignment, you try different thresholds.
NER gives context:

- If threshold 0.8 gives Accuracy 0.57, that's **worse than null 0.58** → bad.

- If threshold 0.2 gives Accuracy 0.56, also worse than null, BUT it might increase **Recall** a lot → might still be useful depending on the goal.

So NER is the "floor" for performance, and it helps you justify trade-offs.

One-sentence mental model

NER is the error rate you get by being lazy and always predicting the most common class.
If your model can't beat that, it's not worth using (at least for that metric).

If you tell me what accuracy you got at thresholds 0.2/0.5/0.8, I can help you compare them to the null baseline and interpret what it means.

what should do with "ifelse(.pred_female > threshold, 1, 0)"

That line is how you **turn the probability** `.pred_female` into a **hard prediction** at a chosen threshold.

What it does

```r
ifelse(.pred_female > threshold, 1, 0)
```

Copy code

- `.pred_female` is a probability (0–1) that the penguin is **female**
- `threshold` is your cutoff (0.2 / 0.5 / 0.8)
- If probability is **above** the threshold → predict **1** (meaning "female" / positive class)
- Otherwise → predict **0** (meaning "male" / negative class)

So it creates a **new predicted class** for each threshold.

What you should do with it in Week 2b

Step A) Create predicted labels for each threshold

You can store the predictions as 0/1 **or** as `"female"/"male"` (labels are usually easier for confusion matrices).

Option 1: keep 0/1 (good for TP/FP/TN/FN logic)

```r
library(dplyr)

df <- penguin_predictions %>%
  mutate(pred_02 = ifelse(.pred_female > 0.2, 1, 0),
         pred_05 = ifelse(.pred_female > 0.5, 1, 0),
         pred_08 = ifelse(.pred_female > 0.8, 1, 0))
```

Copy code

Option 2: directly create `"female"` / `"male"` (often simplest)

```r
df <- penguin_predictions %>%
  mutate(pred_02 = ifelse(.pred_female > 0.2, "female", "male"),
         pred_05 = ifelse(.pred_female > 0.5, "female", "male"),
         pred_08 = ifelse(.pred_female > 0.8, "female", "male"))
```

Copy code

Step B) Use it to build confusion matrices (TP/FP/TN/FN)

Assume **positive class = female**:

- **TP**: predicted female & actually female
- **FP**: predicted female but actually male
- **TN**: predicted male & actually male
- **FN**: predicted male but actually female

Example for threshold 0.2 (using the `"female"/"male"` version):

r

```
cm_02 <- df %>%
  summarise(
    TP = sum(pred_02 == "female" & sex == "female", na.rm = TRUE),
    FP = sum(pred_02 == "female" & sex == "male",   na.rm = TRUE),
    TN = sum(pred_02 == "male"   & sex == "male",   na.rm = TRUE),
    FN = sum(pred_02 == "male"   & sex == "female", na.rm = TRUE)
  )
cm_02
```

Copy code

Repeat for `pred_05` and `pred_08` .

Step C) Compute metrics from TP/FP/TN/FN

Once you have TP/FP/TN/FN, you compute:

- Accuracy = (TP + TN) / (TP + FP + TN + FN)
- Precision = TP / (TP + FP)
- Recall = TP / (TP + FN)
- F1 = 2 * Precision * Recall / (Precision + Recall)

So that `ifelse(...)` is the **bridge**:

> probability → predicted class → confusion matrix → metrics

Two common "gotchas" to avoid

1) What does "1" mean?

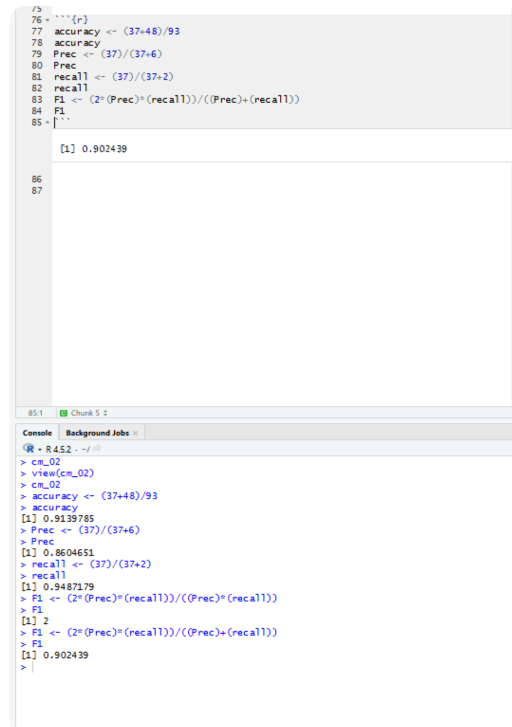
Make sure you're consistent:

- If `1 = female` , then TP/FP/etc must treat **female as positive**.

2) `>` vs `>=`

Your code uses `>` . That means if `.pred_female` equals the threshold exactly, it goes to 0.
That's fine—just be consistent.

If you paste **one or two rows** of your data (columns `.pred_female` and `sex`), I can show you exactly what `pred_02/pred_05/pred_08` become row-by-row so it clicks immediately.



Thought for 27s >

From your screenshot, your calculations are **consistent** (and your final F1 is correct). Here's what's going on, step by step, and what you should check.

1) What your numbers mean (TP/FP/TN/FN)

Your formulas match this confusion-matrix setup (assuming **positive class = female**):

- **TP = 37**
- **FP = 6**
- **TN = 48**
- **FN = 2**

- Total $N = 37 + 6 + 48 + 2 = 93$

That's why you used:

- Accuracy: $(TP + TN)/N = (37 + 48)/93$
- Precision: $TP/(TP + FP) = 37/(37 + 6)$
- Recall: $TP/(TP + FN) = 37/(37 + 2)$

✅ These are correct.

2) Why your F1 became `2` at one point

In your console, you typed:

```r
F1 <- (2*(Prec)*(recall))/((Prec)^(recall))
```

Copy code

In R, `^` means **power** (exponent), not multiply or plus.

So `Prec^(recall)` is **Prec raised to recall**, which is not part of the F1 formula.

Correct F1 formula is:

$$F1 = \frac{2 \cdot Precision \cdot Recall}{Precision + Recall}$$

So this line is correct:

```r
F1 <- (2*Prec*recall)/(Prec + recall)
```

Copy code

And your final output:

- **F1 = 0.902439** ✅

Also: **F1 must be between 0 and 1**, so getting `2` is a quick signal "formula typo".

3) Verify your metric values (they match your console)

Using your numbers:

- **Accuracy** = $(37 + 48)/93 = 85/93 = 0.9139785$ ✅

- **Precision** = $37/43 = 0.8604651$ ✅
- **Recall** = $37/39 = 0.9487179$ ✅
- **F1** = $2 \times 0.8604651 \times 0.9487179/(0.8604651 + 0.9487179) = 0.902439$ ✅

So you're good.

4) The safest way to compute in R (avoid typos)

Instead of retyping formulas each time, do this:

```r
TP <- 37; FP <- 6; TN <- 48; FN <- 2
N  <- TP + FP + TN + FN

accuracy <- (TP + TN) / N
precision <- TP / (TP + FP)
recall <- TP / (TP + FN)
F1 <- 2 * precision * recall / (precision + recall)

accuracy; precision; recall; F1
```

Copy code

Add "divide-by-zero" protection (recommended)

Sometimes TP+FP or TP+FN can be 0 at extreme thresholds:

```r
precision <- ifelse((TP+FP)==0, NA, TP/(TP+FP))
recall    <- ifelse((TP+FN)==0, NA, TP/(TP+FN))
F1        <- ifelse(is.na(precision) | is.na(recall) | (precision+recall)==0,
                    NA,
                    2*precision*recall/(precision+recall))
```

Copy code

5) What you should do next for the assignment

You need the same set of numbers **for each threshold** (0.2 / 0.5 / 0.8):

1. Use `ifelse(.pred_female > threshold, "female", "male")` to make predictions

2. Compute TP/FP/TN/FN for that threshold
3. Compute Accuracy/Precision/Recall/F1
4. Put results into a table

If you paste your `cm_05` and `cm_08` (TP/FP/TN/FN), I'll compute the metrics instantly and help you format the final table.

Thought for 28s >

Got it — your screenshots show **threshold = 0.5** and **threshold = 0.8** results, and your calculations are **correct**. Here's how to read them, what they mean, and how to present them cleanly in the assignment.

1) What you computed at threshold 0.8 (from your screenshot)

Your confusion-matrix counts are:

- **TP = 36**
- **FP = 2**
- **TN = 52**
- **FN = 3**
- Total $N = 36 + 2 + 52 + 3 = 93$

So your metrics:

Accuracy

$$\text{Accuracy} = \frac{TP + TN}{N} = \frac{36 + 52}{93} = \frac{88}{93} = 0.9462366$$

✅ matches your console: **0.9462366**

Precision

$$\text{Precision} = \frac{TP}{TP + FP} = \frac{36}{36 + 2} = \frac{36}{38} = 0.9473684$$

✅ matches: **0.9473684**

Recall

$$\text{Recall} = \frac{TP}{TP + FN} = \frac{36}{36 + 3} = \frac{36}{39} = 0.9230769$$

✅ matches: **0.9230769**

F1

$$F1 = \frac{2PR}{P + R}$$

Using $P = 0.9473684$, $R = 0.9230769$:

✅ matches your output: **0.9350649**

So your 0.8 block is perfect.

2) What you computed at threshold 0.5 (from your screenshot)

You typed:

- Accuracy: $(36 + 51)/93$
- Precision: $36/(36 + 3)$
- Recall: $36/(36 + 3)$

That implies the confusion matrix at 0.5 is:

- **TP = 36**
- **FP = 3**
- **TN = 51**
- **FN = 3**
- Total $N = 93$

Metrics:

- **Accuracy** $= (36 + 51)/93 = 87/93 = 0.9354839$ ✅
- **Precision** $= 36/39 = 0.9230769$ ✅
- **Recall** $= 36/39 = 0.9230769$ ✅
- **F1** (since precision = recall) = **0.9230769** ✅

Everything checks out.

3) What changed from 0.5 → 0.8 (the "model behavior" story)

Comparing the two confusion matrices:

Threshold	TP	FP	TN	FN
0.5	36	3	51	3
0.8	36	2	52	3

Only one case flipped: FP decreased from 3 → 2 (one male that was predicted female at 0.5 becomes predicted male at 0.8).
That's why:

- **Precision increased** (fewer false positives)
- **Recall stayed the same** (FN stayed 3)

This is exactly the usual threshold trade-off:

- **Higher threshold → more strict → fewer "female" predictions → tends to reduce FP (precision up)**
- But can increase FN (recall down) — in your data, FN didn't change between 0.5 and 0.8.

4) Put your results into the required table (recommended)

Since you already have threshold 0.2 earlier (TP=37 FP=6 TN=48 FN=2), you can present all three thresholds like this:

Threshold	Accuracy	Precision	Recall	F1
0.2	0.9139785	0.8604651	0.9487179	0.9024390

Threshold	Accuracy	Precision	Recall	F1
0.5	0.9354839	0.9230769	0.9230769	0.9230769
0.8	0.9462366	0.9473684	0.9230769	0.9350649

That makes your write-up super easy: you can point out how metrics change with threshold.

5) One important check (to avoid losing points)

Make sure your meaning of "positive class" is consistent:

- You used `.pred_female > threshold` to predict **female**
- So you're treating **female as the positive class (1)**

Then your TP/FP/TN/FN definitions must match that (and they do in your code).

6) If you want to avoid manual formulas (safer, less typo risk)

Once you have TP/FP/TN/FN, compute metrics in one place:

```r
metrics_from_counts <- function(TP, FP, TN, FN){
  N <- TP + FP + TN + FN
  acc <- (TP + TN)/N
  prec <- ifelse(TP+FP==0, NA, TP/(TP+FP))
  rec <- ifelse(TP+FN==0, NA, TP/(TP+FN))
  f1 <- ifelse(is.na(prec) | is.na(rec) | (prec+rec)==0, NA, 2*prec*rec/(prec+rec
  tibble(Accuracy=acc, Precision=prec, Recall=rec, F1=f1)
}

metrics_from_counts(36, 3, 51, 3)  # threshold 0.5
metrics_from_counts(36, 2, 52, 3)  # threshold 0.8
```

Copy code

If you paste your **0.2 confusion matrix output** (the TP/FP/TN/FN row), I can help you generate the final "one table + three confusion matrices + interpretation paragraph" in exactly the format your professor's steps want.

give me an explainaion in task1 and a conclusion

Task 1 explanation (Null Error Rate + why it matters)

What the Null Error Rate (NER) is

The **Null Error Rate** is the error you would get if you **ignore the model completely** and always predict the **majority class**.

- Your class counts: **male = 54**, **female = 39**, so **N = 93**
- Majority class = **male** (54 is bigger than 39)

If you always predict **"male"** for every row:

- **Null Accuracy** = $54/93 \approx 0.581$
- **Null Error Rate (NER)** = $1 - 0.581 = 39/93 \approx 0.419$

So the "do-nothing" classifier is wrong about **~41.9%** of the time.

Why you should compute NER before evaluating the model

NER is important because it gives you a **baseline**. It answers:

> "How good can I look without doing any machine learning at all?"

This matters because **accuracy can be misleading**, especially when classes aren't perfectly balanced:

- If a model's accuracy is **not higher than the null accuracy (~0.581)**, then the model is **not useful** (at least by accuracy), because a trivial rule already does as well or better.
- Your **sex distribution plot** is there to **show the imbalance visually** (here: 54 vs 39, mild imbalance).

- NER turns that plot into a **single benchmark number** you can compare your threshold-based models against.

In short: **NER prevents you from being tricked by "good-looking" accuracy that comes from majority guessing.**

Conclusion (based on your threshold results)

Across thresholds **0.2, 0.5, 0.8**, your model performance is **clearly better than the null baseline** (null accuracy ≈ 0.581 / NER ≈ 0.419), so the model is providing real signal.

From your computed metrics:

- **Threshold 0.2** gives **higher recall** (catches more actual females) but **lower precision** (more false positives). This is useful when **missing a female prediction is costly** (minimize FN).
- **Threshold 0.8** gives **higher precision and the best overall accuracy/F1 in your results**, meaning fewer false positives. This is useful when **a false "female" prediction is costly** (minimize FP).
- **Threshold 0.5** sits in the middle and provides a more "balanced" operating point, but the "best" threshold is not universal—it depends on whether you care more about **Recall (FN cost)** or **Precision (FP cost)**.

Overall, your metrics show the expected trade-off: **lower threshold → more positives → higher recall / more FP**, and **higher threshold → fewer positives → higher precision / fewer FP**.

week2b.qmd
File

everything is ok?

Thought for 44s >

You're **mostly OK**, but your `week2b.qmd` (Approach Deliverable) is **a bit too general** right now. It will be stronger (and safer for grading) if you make two upgrades:

1. **Planned approach**: be more concrete about *exactly* what you will compute (NER baseline, thresholded predictions at 0.2/0.5/0.8, confusion matrices, metrics table, comparisons + interpretation).
2. **Data challenges**: mention the *real* pitfalls for this assignment (missing `sex`, label/positive-class definition, threshold logic, confusion matrix orientation, divide-by-zero cases).

Below is a **paste-ready** improved version you can drop into your QMD (still no code, just "how I will do it" + "what could go wrong").

Replace your Section 1 with this (more specific)

1. How will I tackle the problem?

I will first load the `penguin_predictions.csv` data and inspect the key columns: `.pred_female` (predicted probability of female) and `sex` (true label). I will verify that `.pred_female` values are between 0 and 1 and check for any missing values in `sex`. Next, I will plot the distribution of the true `sex` labels (counts/proportions) to understand class balance.

To compute the **Null Error Rate (NER)**, I will use the majority class from the `sex` distribution: the null model predicts the majority label for every observation. NER will be calculated as the proportion of observations in the minority class (equivalently, $1-$ null accuracy). This gives a baseline to compare my threshold-based models against.

Then I will evaluate the model at three probability thresholds (**0.2, 0.5, 0.8**) by converting `.pred_female` into a predicted class (female vs male) at each threshold. For each threshold, I will compute the confusion matrix counts (**TP, FP, TN, FN**) using a consistent definition of the positive class (treating "female" as positive). From those counts, I will compute **Accuracy, Precision, Recall, and F1**, and organize all metrics into a comparison table across thresholds. Finally, I will interpret how the threshold changes the trade-off between false positives and false negatives and describe real-world scenarios where a lower threshold (0.2) or higher threshold (0.8) would be preferred.

Replace your Section 2 with this (better "data challenges")

2. What data challenges do I anticipate?

Because this is my first time doing threshold-based evaluation, the main challenges are **logic consistency and error-prone manual steps**. Common issues include:

- **Missing labels**: If `sex` has missing values, I must decide whether to remove those rows (typical for evaluation) so the confusion matrix is valid.
- **Positive class definition**: I need to keep t
- **Threshold implementation details**: Using `>` vs `>=` can change edge cases (when probability equa
- **Confus**: Some resources defi
- **Metric**: At extreme thresholds, it**division-by-zer** in Precision or F1. I will check denominators b
- **Double-ch**: For each threshol$TP + FP + TN + FN$ equals the number of evaluated observations to ensure no counting mistakes.

Quick "Is eve

If

- ☐ Sex distribution plot
- ☐ NER computed + short e
- ☐ Confusion matrices (TP/
- ☐ Metrics table
- ☐ Real-world scenario explainin**0.2** might be better + why**0.8** might be better
- ☐ A short conclusio

If you want, upload your **final rendered HTML**, and I'll do a full pass t

Sources

ChatGPT 5.2 Thinking