

If you want, I can **write the exact R code next** to start computing the **null error rate** and **plotting the class distribution** for your CSV. Do you want me to do that?



when i click on text(readr) it say namespace rlang 1.1.6 already loaded but rlang 1.1.7 is required

Ah! That message happens because your version of `rlang` is outdated. RStudio is telling you that `readr` **(the import tool)** needs a newer version of `rlang`. We can fix it quickly.

Here's how:

---

### Step 1: Update `rlang`

```
r
```

Copy code

```
install.packages("rlang")
```

- This will download the latest version (1.1.7 or higher).
- After it installs, restart RStudio (**Session → Restart R**) to make sure it loads the new version.

---

### Step 2: Load `readr` (optional)

```
r
```

Copy code

```
library(readr)
```

Now the **Import Dataset → From Text (readr)** should work without errors.

---

### Step 3: Load your CSV directly with `read.csv` (bypasses `readr` entirely)

If you want a quick solution without worrying about `readr` :

```
r
```

[Copy code](#)

```
penguins <- read.csv("C:/Users/YourName/Downloads/penguin_predictions.csv")  
head(penguins)
```

This works even if `readr` isn't loading correctly.

---

💡 **Tip:** Using `read.csv` is simpler for this assignment — you don't need the fancy import tool unless you want a GUI preview.

---

I can give you the **exact code next** to compute the **null error rate and class distribution plot** directly using your CSV. Do you want me to do that?

i loaded it but i have it on my work space how do i get on the bottom right panel with other files

Ah, I see exactly what you mean! In RStudio, the **bottom-right panel** is usually the **Environment/History/Connections/Files/Plots/Packages/Help/Viewer** tabs.

When you "load" a CSV with `read.csv()`, the data appears in your **Environment tab** (top-right by default), but it isn't automatically listed as a file in the **Files tab** (bottom-right) — that panel shows **actual files on your computer**, not R objects.

Here's how to get what you want:

---

## Option 1: Save the data frame to an RDS file

This will create a file you can see in the **Files tab**:

```
r
```

[Copy code](#)

```
saveRDS(penguins, file = "penguins.rds")
```

- Now in the **Files tab**, you should see `penguins.rds`.
- Later, you can reload it with:

```
r
```

[Copy code](#)

```
penguins <- readRDS("penguins.rds")
```

---

## Option 2: Save as CSV in your working directory

```
r
```

[Copy code](#)

```
write.csv(penguins, "penguins_saved.csv", row.names = FALSE)
```

- Now `penguins_saved.csv` appears in the **Files tab**.
- You can click on it in RStudio to open a spreadsheet-style viewer.

---

## Option 3: View the data frame directly

You can also open the loaded CSV in a table view:

```
r
```

[Copy code](#)

```
View(penguins)
```

- This opens the data in a new tab inside RStudio (like Excel), so you can scroll through it.

---

✅ **Key point:** The **Files panel** shows actual files, not objects. Objects like `penguins` are in the **Environment panel**. Using `View()` or saving to CSV/RDS lets you interact with them like files.

If you want, I can give you the **exact sequence of R commands** to load your CSV, check null error rate, and plot the class distribution — all ready to run. That way your data is fully visible and usable.

Do you want me to do that?



so if i do that will i see read.csv or something else like write.csv

Ah, I get your confusion! Let me clarify clearly.

---

### Key distinction: R objects vs files in Files tab

- `read.csv("penguin_predictions.csv")` → **loads the CSV into R as a data frame** (e.g., `penguins` ).
  - You **won't see** `read.csv` **in the Files tab** — the Files tab shows **actual files on your computer**, not the commands you ran.
  - The data now exists in your **Environment panel** (top-right), under the name `penguins` .
- `write.csv(penguins, "penguins_saved.csv")` → **creates a new CSV file** on your disk.
  - This **new file** will appear in the **Files tab** at the bottom-right.
  - You can now click it to open it, see it in spreadsheet style, or load it later with `read.csv` .

---

### Simple way to see your CSV as a file in Files tab

r

 Copy code

```
# Save the loaded data frame as a new CSV in your working directory
write.csv(penguins, "penguins_saved.csv", row.names = FALSE)
```

- After running this, go to **Files tab** → you will see `penguins_saved.csv` .
- Double-click to open it in RStudio’s viewer.

💡 **Summary:**

Action	Appears in Files tab?	Appears in Environment?
<code>read.csv()</code>	❌ No	✅ Yes ( <code>penguins</code> )
<code>write.csv()</code>	✅ Yes ( <code>penguins_saved.csv</code> )	❌ No (creates file on disk)
<code>View(penguins)</code>	❌ No	✅ Opens a table viewer

If you want, I can give you a **ready-to-run script** that:

1. Loads your CSV
2. Plots class distribution
3. Calculates null error rate

...so you can start the assignment right away.

Do you want me to do that?



which one is best `read.csv` or `write.csv`

Ah, now I understand exactly what you’re asking! 😊

It's not really a matter of "which is best" — **they do completely different things**. Think of them like opposite sides of the same coin:

---

### 1 `read.csv()`

- **Purpose:** Load a CSV file **from your computer into R**.
- **Use case:** When you want to **work with data** in R (analyze it, make plots, run models).
- **Output:** Creates a **data frame** in your R Environment (e.g., `penguins`).
- **Example:**

```
r
```

[Copy code](#)

```
penguins <- read.csv("penguin_predictions.csv")
```

---

### 2 `write.csv()`

- **Purpose:** Save a data frame **from R back to a CSV file on your computer**.
- **Use case:** When you've modified data in R and want to **export it** for later use, or to share.
- **Output:** Creates a **file on disk**, visible in the Files tab.
- **Example:**

```
r
```

[Copy code](#)

```
write.csv(penguins, "penguins_saved.csv", row.names = FALSE)
```

---

### ✓ Rule of thumb for your assignment:

- To **start working** with the CSV: use `read.csv()`.
- Only use `write.csv()` if you want to **save or share** your processed data.

So for **loading your** `penguin_predictions.csv` **and doing the analysis**, `read.csv()` is what you need.

---

If you want, I can now give you the **exact R code to:**

1. Load your CSV with `read.csv()`
2. Plot the sex distribution
3. Compute the null error rate

...so you can start your assignment immediately.

Do you want me to do that?



i am struggling to get the path of  
penguins\_prediction.csv file on my pc

No worries — this is a super common issue in RStudio! Let's solve it step by step so you can **load your CSV correctly**.

---

## Step 1: Find the full path of your CSV

1. Locate the file on your computer (e.g., in **Downloads** or Desktop).
2. Right-click on the file → **Properties** (Windows) or **Get Info** (Mac).
3. Copy the **full path**, including the file name.

Examples:

- Windows:

```
swift
```

Copy code

```
C:/Users/Guibril/Downloads/penguin_predictions.csv
```

- Mac:

```
swift
```

Copy code

```
/Users/Guibril/Downloads/penguin_predictions.csv
```

⚠ Important: Use **forward slashes** / in R, even on Windows.

C:\Users\Guibril\Downloads\penguin\_predictions.csv → R will give errors.

---

## Step 2: Set your working directory (optional but easier)

You can set RStudio to look in the folder where your CSV is:

```
r
```

 Copy code

```
# Windows example  
setwd("C:/Users/Guibril/Downloads")
```

```
# Mac example  
setwd("/Users/Guibril/Downloads")
```

- After this, you **only need the file name** in `read.csv()` :

```
r
```

 Copy code

```
penguins <- read.csv("penguin_predictions.csv")
```

- Check it loaded correctly:

```
r
```

 Copy code

```
head(penguins)  
str(penguins)
```

---

## Step 3: Alternative: Use RStudio GUI

1. In RStudio, go to **Files tab** → **navigate to the folder** containing the CSV.
  2. Click the file → choose **"Import Dataset"** → **From Text (readr)**.
  3. RStudio automatically fills the path for you.
- 

✅ **Quick tip:** If you're not sure about the exact path, the easiest is:

```
r
```

 Copy code



```
# Opens a file chooser dialog
penguins <- read.csv(file.choose())
```

- This pops up a window where you can **click your CSV**, and RStudio will automatically get the path.

---

If you want, I can now give you a **ready-to-run R script** that loads the CSV (using `file.choose()` to avoid path issues), plots the class distribution, and calculates the null error rate — so you can continue your assignment immediately.

Do you want me to do that?



i think it saved in number

Ah! I think I understand — your file might have been **renamed automatically by your system** when saving from the browser, like:

scss

 Copy code

```
penguin_predictions (1).csv
penguin_predictions.csv
penguin_predictions(2).csv
```

or sometimes the file name is just a number like:

 Copy code

```
12345.csv
```

This happens sometimes with downloads.

---

## How to fix it

1. Check the Downloads folder (or wherever you saved it).
2. Look for any CSV files that match the size or date you downloaded.
3. Rename it to something simple, e.g.:

 Copy code

```
penguin_predictions.csv
```

Use only letters, underscores, or numbers. Avoid spaces or parentheses for simplicity.

---

## Load it in R

Once you rename it:

```
r
```

 Copy code

```
# If you know the folder
setwd("C:/Users/Guibril/Downloads") # Adjust for your computer
penguins <- read.csv("penguin_predictions.csv")
```

```
# OR just choose it manually
penguins <- read.csv(file.choose()) # pops up a file chooser
```


- Check it worked:

```
r
```

 Copy code

```
head(penguins)
str(penguins)
```

---

 **Tip:** Using `file.choose()` is easiest if you're unsure about the exact file name — you just click the file and R will handle the path automatically.

---

If you want, I can now give you the **full R code to start the assignment**: load the CSV, check class distribution, and calculate null error rate — **ready to run even if your file name is a number**.

Do you want me to do that?



i moved it to desktop but get info do not gives me the paf  
i see <https://github.com>

Ah! I see exactly what's happening — you're confusing **where the file is on your computer** with the **download link on GitHub**.

When you click "Get Info" or look at GitHub, it might show something like:

arduino

 Copy code

```
https://github.com/.../penguin_predictions.csv
```

That **is not the path on your computer** — that's the online location. R **cannot load files directly from GitHub** unless you use a **URL** and read it in a **special way**.

Since you **moved it to your Desktop**, you need the **local file path**. Here's how to get it correctly:

---

## Step 1: Find the path on your Desktop

### Windows

1. Go to your Desktop and find the file ( `penguin_predictions.csv` ).
2. Hold **Shift**, then **Right-click** → **Copy as path**.
  - It will give something like:

arduino

 Copy code

```
"C:\\Users\\Guibril\\Desktop\\penguin_predictions.csv"
```

3. In R, replace backslashes `\` with forward slashes `/` :

```
r
```

[Copy code](#)

```
penguins <- read.csv("C:/Users/Guibril/Desktop/penguin_predictions.csv")
```

---

## Mac

1. Go to your Desktop, right-click the file → **Get Info**.
2. Look for **"Where"** to see the folder path (e.g., `/Users/Guibril/Desktop` ).
3. Combine with the file name:

```
r
```

[Copy code](#)

```
penguins <- read.csv("/Users/Guibril/Desktop/penguin_predictions.csv")
```

---

## Step 2: Alternative — use `file.choose()` (easiest)

```
r
```

[Copy code](#)

```
penguins <- read.csv(file.choose())
```

- This will **open a file picker**.
  - Navigate to your Desktop, click the CSV, and RStudio will load it automatically.
  - You don't have to worry about paths at all.
- 

✅ **Tip:** For beginners, I recommend **using** `file.choose()` . It avoids any path mistakes.

---

If you want, I can now give you a **ready-to-run R script** that:

1. Loads the CSV (using `file.choose()` so paths don't matter)
2. Plots class distribution
3. Computes null error rate

...so you can start your assignment immediately.

Do you want me to do that?



well it works now i just used the github link  
`https://github.com/acadlin/data/blob/master/penguin_predictions.csv`

Ah perfect! That makes sense — you're loading the CSV **directly from GitHub**. ✅

One thing to note: the link you just gave  
( `https://github.com/acadlin/data/blob/master/penguin_predictions.csv` ) is the **GitHub webpage for the CSV**, not the raw CSV file itself. R needs the **raw file URL** to read it correctly.

---

### Step 1: Get the raw CSV URL

1. Go to your GitHub link in a browser.
2. Click **"Raw"** (top-right of the file preview).
3. The URL should now look like:

arduino

 Copy code

`https://raw.githubusercontent.com/acadlin/data/master/penguin_predictions.`

---

### Step 2: Read the CSV in R directly from GitHub

r

 Copy code

`penguins <- read.csv("https://raw.githubusercontent.com/acadlin/data/maste`

```
# Check it loaded correctly
```

```
head(penguins)
```

```
str(penguins)
```

✅ Now you have the CSV **directly in your R Environment** without downloading anything manually.

---

If you want, I can write the **next part of your assignment** — null error rate, class distribution plot, and everything ready to go — using this URL so you don't have to worry about file paths at all.

Do you want me to do that?



```
can you analyse and correct and make it professional ---  
title: "Penguins_predictions"  
name: Guibril Ramde  
date: 2026-02-05  
output: html_document  
---
```

# Approach

on this assignement i will first donwload the raw github  
link of the csv file and load it on R.  
will need to install required library  
then i will proceed to calculate the new erro rate.  
then proceed to the plot that will give me a baseline of  
my data model  
then i can now proceed to calculate the confusion matrix  
and analyse it.  
present the matrix in a table and the use case.

[https://raw.githubusercontent.com/acadlin/data/refs/head/s/master/penguin\\_predictions.csv](https://raw.githubusercontent.com/acadlin/data/refs/head/s/master/penguin_predictions.csv)

Absolutely! I can take your draft and turn it into a **professional, polished R Markdown document** that is clear, grammatically correct, and properly structured for your assignment. I'll also make it **ready to render as HTML**.

Here's a revised version:

yaml

 Copy code

```
---  
title: "Penguin Predictions Analysis"  
author: "Guibril Ramde"  
date: "2026-02-05"  
output: html_document  
---
```

*# Approach*

In this assignment, I will analyze the performance of a binary classificat

The workflow will be as follows:

1. **Load the data**: I will download the raw CSV file from GitHub and loa
2. **Install and load required libraries**: Necessary libraries such as `g
3. **Compute the null error rate**: This baseline metric shows the error i
4. **Visualize class distribution**: A plot will provide insight into the
5. **Compute confusion matrices**: I will calculate TP, FP, TN, and FN at
6. **Calculate performance metrics**: Accuracy, precision, recall, and F1
7. **Discuss threshold use cases**: I will describe scenarios where differ

The raw CSV file is located at:

[Penguin Predictions CSV](<https://raw.githubusercontent.com/acadlin/data/r>

---

 **Improvements made:**

1. Corrected **spelling, grammar, and capitalization**.
  2. Structured your workflow with **numbered steps** — makes it look professional and organized.
  3. Used **clear, academic language** instead of casual phrasing.
  4. Formatted the GitHub link properly.
- 

If you want, I can **take this a step further** and add the **full R code chunks** right in the R Markdown to:

- Load the CSV from GitHub
- Compute null error rate
- Plot class distribution
- Compute confusion matrices and metrics

...so your R Markdown is **fully ready to knit to HTML**.

Do you want me to do that next?



```
from this plot how do i get the table to be from high to
low ? since male is high percentage and woman low
percentage and how do i get the percentage written on
top of the each plot ggplot(penguins, aes(x = sex, fill =
sex)) +
  geom_bar() +
  labs(title = "Class Distribution of Sex", x = "Sex", y =
"Count") +
  theme_minimal()
```

Great question — this is exactly the kind of polish professors like to see 🍌

You're very close already. We'll do **two things**:

1. **Order the bars from highest to lowest frequency**



## 2. Add percentage labels on top of each bar

---

### 1 Order the bars from high → low

`ggplot()` plots categories in the order of the factor levels.

So we first **reorder** `sex` **by count**.

```
r  
  
library(dplyr)  
library(ggplot2)  
  
penguins %>%  
  count(sex) %>%  
  mutate(sex = reorder(sex, n)) %>%  
  ggplot(aes(x = sex, y = n, fill = sex)) +  
  geom_col()
```

[Copy code](#)

But we still need **percentages** and **labels**, so let's do it cleanly in one pipeline.

---

### 2 Add percentages on top of each bar

Here is the **professional, final version** you should use:

```
r  
  
penguins %>%  
  count(sex) %>%  
  mutate(  
    percent = n / sum(n) * 100,  
    sex = reorder(sex, -percent)  # reorder high → low  
  ) %>%  
  ggplot(aes(x = sex, y = percent, fill = sex)) +  
  geom_col() +  
  geom_text(  
    aes(label = paste0(round(percent, 1), "%")),  
    vjust = -0.3,  
    size = 5  
  ) +
```

[Copy code](#)

```
labs(
  title = "Class Distribution of Sex",
  x = "Sex",
  y = "Percentage"
) +
theme_minimal() +
theme(legend.position = "none")
```

### ✅ What this does (important for understanding)

- `count(sex)` → creates a table of counts
- `percent = n / sum(n) * 100` → converts counts to percentages
- `reorder(sex, -percent)` → sorts bars **from highest to lowest**
- `geom_col()` → bar chart using computed values
- `geom_text()` → writes the percentage **on top of each bar**



### Result

- **Male** appears first (highest percentage)
- **Female** appears second (lower percentage)
- Percentages are clearly labeled on top of bars
- Plot is clean, professional, and publication-ready



### Optional (extra polish)

If you want the bars horizontal (sometimes easier to read):

```
r
```

Copy code

```
+ coord_flip()
```

If you want, next I can help you **turn this plot explanation into a short paragraph** that you can paste directly into your assignment under "Null Error Rate / Class Distribution".



Error in compute\_metrics(penguins, t) : object 'recall' not found. #recomputing predicted class using threshold

```
compute_metrics = function(data, threshold) {
  data$pred_class_thresh = ifelse(data$pred_female >
threshold, 1, 0)

  #Confusion matrix
  TP = sum(data$pred_class_thresh == 1 & data$sex ==
1)
  FP = sum(data$pred_class_thresh == 1 & data$sex ==
0)
  TN = sum(data$pred_class_thresh == 0 & data$sex ==
0)
  FN = sum(data$pred_class_thresh == 0 & data$sex ==
1)

  #Performance metrics
  accuracy = (TP + TN) / nrow(data)
  precision = ifelse((TP + FP) == 0, NA, TP / (TP + FN))
  f1 = ifelse(is.na(precision) | is.na(recall) | (precision +
recall) ==
                2 * precision * recall / (precision + recall))
  list(
    ConfusionMatrix = matrix(c(TP, FP, FN, TN),
nrow = 2,
                                dimnames = list("Predicted" =
c("1", "0"),
                                                "Actual" = c("1", "0"))),
    Accuracy = accuracy,
    Precision = precision,
    Recall = recall,
    F1 = f1
  )
}
```

```
thresholds = c(0.2, 0.5, 0.8)
results = lapply(thresholds, function(t)
  compute_metrics(penguins, t))
names(results) = paste0("Threshold_", thresholds)
results
```

You're **very close** — this error is happening for two clear reasons, and once you see them, it'll click immediately 🍌

Let's walk through it cleanly and fix it **properly and professionally**.

## 🔴 Why you're getting this error

### Error message

```
pgsql
```

 Copy code

```
object 'recall' not found
```

### Problem 1: recall is never defined

You **use** `recall` in the F1 calculation, but you never **created** it.

You need:

```
r
```

 Copy code

```
recall = TP / (TP + FN)
```

### Problem 2: Broken `ifelse()` for F1

This part is malformed:

```
r
```

 Copy code

```
f1 = ifelse(is.na(precision) | is.na(recall) | (precision + recall) ==
           2 * precision * recall / (precision + recall))
```

Issues:

- `ifelse()` is missing its **TRUE** and **FALSE** values
- Parentheses are mismatched
- The formula is placed inside the condition

## ✅ Correct, clean, PROFESSIONAL version

Here is a **fully corrected and robust function** you should use:

r

 Copy code

```
compute_metrics <- function(data, threshold) {

  # Recompute predicted class using threshold
  data$pred_class_thresh <- ifelse(data$.pred_female > threshold, 1, 0)

  # Confusion matrix components
  TP <- sum(data$pred_class_thresh == 1 & data$sex == 1)
  FP <- sum(data$pred_class_thresh == 1 & data$sex == 0)
  TN <- sum(data$pred_class_thresh == 0 & data$sex == 0)
  FN <- sum(data$pred_class_thresh == 0 & data$sex == 1)

  # Performance metrics
  accuracy <- (TP + TN) / nrow(data)

  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)
  )

  list(
```


```

ConfusionMatrix = matrix(
  c(TP, FP, FN, TN),
  nrow = 2,
  dimnames = list(
    Predicted = c("Positive", "Negative"),
    Actual = c("Positive", "Negative")
  )
),
Accuracy = accuracy,
Precision = precision,
Recall = recall,
F1 = f1
)
}

```

## Run it for multiple thresholds

```

r
                                                                     Copy code






thresholds <- c(0.2, 0.5, 0.8)

results <- lapply(thresholds, function(t) compute_metrics(penguins, t))
names(results) <- paste0("Threshold_", thresholds)

results

```

## Why this version is correct (and assignment-ready)

-  Explicit TP / FP / TN / FN logic
-  Recall is correctly defined
-  F1 score is computed safely (no divide-by-zero)
-  Output is structured and readable
-  Matches your assignment's **manual confusion-matrix-first philosophy**

## Next step (optional but recommended)

I can help you:

- Turn results into a **clean comparison table**
- Write a **short interpretation paragraph** for each threshold
- Match your professor's wording exactly

Just tell me what you want next 🙌



why do i need example of threshold why can i get it from the penguins data i got

This is a **really good question**, and asking it shows you're actually thinking like a data scientist instead of just following instructions. 🙌

Let's clear up the confusion carefully.

---

## The short, honest answer

You **cannot** get "real-world threshold use cases" from the penguin dataset itself.

The penguin data only tells you:

- probabilities
- predictions
- true labels
- how metrics change when the threshold changes

It **does NOT** tell you the consequences of mistakes in the real world.

That's why your professor asks for **examples**.

---

## What the penguin data *can* tell you

From the penguin data, you *can* observe things like:

- Lower threshold → more predicted females
- Higher threshold → fewer predicted females
- Recall increases as threshold decreases
- Precision increases as threshold increases

That's **model behavior**, not **decision context**.

The dataset stops here.

---

## What the assignment is testing (this is key)

Your professor is testing whether you understand that:

**There is no "best" threshold without context.**

Threshold choice depends on:

- Cost of false positives
- Cost of false negatives
- Resources
- Risk tolerance

Penguins don't have consequences.

**Real decisions do.**

So they force you to *leave the dataset* and think conceptually.

---

## Why you're asked for examples like 0.2 and 0.8

### Threshold = 0.2

- High recall
- Many positives predicted
- Many false positives

This is good when:

- Missing a positive is very costly

Example:

- Medical screening



- Fraud detection pre-filter
  - Security alerts
- 

## Threshold = 0.8

- High precision
- Few positives predicted
- Few false positives

This is good when:

- False positives are costly

Example:

- Manual investigations
  - Legal decisions
  - Expensive follow-ups
- 

## How to write this professionally (use this wording)

You can literally say:

*The penguin dataset allows us to observe how performance metrics change as the probability threshold varies. However, the dataset itself does not provide information about the real-world costs associated with false positives and false negatives. Therefore, threshold selection must be guided by external context rather than the data alone.*

That sentence alone is **A-level reasoning**.

---

## Bottom line (important)

- ❌ The dataset does **not** contain threshold "use cases"
- ✅ The dataset shows **trade-offs**
- ✅ Use cases come from **decision context**, not data
- ✅ That's exactly why your professor asked for them

If you want, I can help you:

- Write the **threshold explanation section** perfectly
- Tie it back explicitly to your penguin results
- Make it sound like a grad-level ML report

Just say the word.



base on this result why with different tht=reshold we get the same prediction and accuracy same recall and f1

This is an **excellent observation**, and it's exactly the kind of thing instructors *want* you to notice. Nothing is "wrong" with your code or your understanding — this can **legitimately happen**. Let's unpack **why** in a clean, data-science way.

---

## What you're observing

You changed the threshold (e.g., 0.2, 0.5, 0.8), but you're seeing:

- Same predictions
- Same confusion matrix
- Same accuracy, recall, and F1

That feels counterintuitive... but it has a **very specific reason**.

---

## The core reason (this is the key insight)

Changing the threshold only changes predictions if there are predicted probabilities between those thresholds.

If *all* predicted probabilities are either:

- **below the smallest threshold**, or
- **above the largest threshold**

then changing the threshold does **nothing**.

### Concrete example

Imagine your `.pred_female` values look like this:

 Copy code

```
0.02, 0.05, 0.91, 0.94, 0.97
```

Now test thresholds:

Threshold	Predicted class
0.2	same
0.5	same
0.8	same

Why?

- All values < 0.2 stay negative
- All values > 0.8 stay positive
- **No probabilities fall in (0.2, 0.8)**

 So predictions never flip.

### How to verify this in *your* data

Run this:

```
r  
  
summary(penguins$.pred_female)
```

 Copy code

Then check:

```
r
```

 Copy code

```
sum(penguins$.pred_female > 0.2 & penguins$.pred_female <= 0.8)
```

If this returns **0** or a **very small number**, that explains everything.

---

## Why accuracy, recall, and F1 are identical

All performance metrics are derived from:

- TP
- FP
- TN
- FN

If the **confusion matrix does not change**, then:

- Accuracy stays the same
- Recall stays the same
- F1 stays the same

This isn't a bug — it's **mathematically inevitable**.

---

## What this tells you about the model

This is actually a **strong insight**:

- The model is making **very confident predictions**
- Probabilities are clustered near 0 or 1
- The model is **insensitive to moderate threshold changes**

This can mean:

- The model separates classes well
- or**
- The model is overconfident (poorly calibrated)

Both are valid discussion points.

---

## How to explain this professionally (use this!)

You can write:

*The identical performance metrics across multiple thresholds indicate that the predicted probabilities are concentrated away from the threshold boundaries. As a result, changing the threshold does not alter the predicted class labels, leading to identical confusion matrices and derived metrics.*

That sentence alone is **top-tier analysis**.

---

## Want to go one level deeper? (optional but impressive)

Plot the probability distribution:

r

 Copy code

```
ggplot(penguins, aes(x = .pred_female)) +  
  geom_histogram(bins = 30, fill = "steelblue", color = "white") +  
  labs(  
    title = "Distribution of Predicted Probabilities",  
    x = "Predicted Probability of Female",  
    y = "Count"  
  ) +  
  theme_minimal()
```

If you see spikes near 0 and 1 → your explanation is confirmed visually.

---

## Bottom line

- Same metrics across thresholds = **no probabilities crossing thresholds**
- This is **expected behavior**, not an error
- You spotted a subtle but important modeling insight

If you want, I can help you turn this into a **short paragraph** perfectly tailored for your assignment submission.

can you rewrite it more well organized #Conclusion

this assignment 2B was challenging and required more focus and understanding of the subject.

reproducing the threshold and the confusion matrix was challenging.

in overall The identical performance metrics across multiple thresholds indicate that the predicted probabilities are concentrated away from the threshold boundaries. As a result, changing the threshold does not alter the predicted class labels, leading to identical confusion matrices and derived metrics.

Absolutely — here's a **clean, well-organized, professional conclusion** that keeps your meaning but sounds polished and academic.

You can paste this directly into your assignment.

---

## Conclusion

Assignment 2B was challenging and required careful attention to both the conceptual and practical aspects of classification model evaluation. In particular, reproducing predicted classes at different probability thresholds and manually computing the corresponding confusion matrices required a deeper understanding of how model outputs translate into performance metrics.

Overall, the identical performance metrics observed across multiple thresholds indicate that the model's predicted probabilities are concentrated away from the threshold boundaries. As a result, adjusting the probability threshold does not change the predicted class labels, leading to identical confusion matrices and, consequently, identical accuracy, recall, and F1 scores. This outcome highlights the importance of examining the distribution of predicted probabilities when interpreting threshold-based evaluation results.

If you want, I can also:

- Slightly **simplify the language** if your class prefers plainer writing, or
- Make it sound **more advanced/graduate-level**, or
- Help you connect this conclusion back to the **null error rate baseline**.

Just tell me 👍



how to use quarto

Great question — **Quarto is worth learning**, and if you already understand R Markdown, you're 80% there. Let's walk through it **clearly and practically**, no fluff.

## What is Quarto (in one sentence)

**Quarto** is a modern document system that lets you combine **code + text + output** to create HTML, PDF, or Word reports — like R Markdown, but cleaner and more powerful.

### 1 Check if Quarto is installed

In RStudio, open the **Console** and run:

```
r
```

 Copy code

```
quarto --version
```

- If you see a version number → you're good.
- If not, download it from: <https://quarto.org/download> ↗

👉 Restart RStudio after installing.

## 2 Create a Quarto document in RStudio

1. **File** → **New File** → **Quarto Document**
2. Choose:
  - **Title:** e.g. Penguin Predictions Analysis
  - **Author:** Guibril Ramde
  - **Format:** HTML
3. Click **Create**

This creates a file like:

 Copy code

```
analysis.qmd
```

---

## 3 Quarto file structure (very important)

A `.qmd` file has three main parts:

### ◆ YAML header (top)

```
yaml
```

 Copy code

```
---  
title: "Penguin Predictions Analysis"  
author: "Guibril Ramde"  
date: "2026-02-05"  
format: html  
---
```

Quarto uses `format:` instead of `output:` (R Markdown difference).

---

### ◆ Markdown text

Just write normally:

```
markdown
```

 Copy code



## ## Approach

This report analyzes the performance of a binary classification model pred

---

### ◆ Code chunks (same idea as R Markdown)

```
{r}
```

[Copy code](#)

```
penguins <- read.csv("https://raw.githubusercontent.com/acatlin/data/master/head(penguins)
```

💡 No backticks needed in the editor — Quarto handles it cleanly.

---

### 4 Render (this replaces “Knit”)

To generate the HTML:

- Click **Render** (top of the editor)
- OR run:

```
r
```

[Copy code](#)

```
quarto render analysis.qmd
```

This creates:

[Copy code](#)

```
analysis.html
```

---

### 5 Example: Minimal Quarto workflow (copy–paste)

```
yaml
```

[Copy code](#)

```
---  
title: "Penguin Predictions Analysis"
```

```
author: "Guibril Ramde"
```

```
format: html
```

```
---
```

```
## Load Data
```

```
`r`
```

```
penguins <- read.csv("https://raw.githubusercontent.com/acatlin/data/master/penguins")
```

## Class Distribution

```
{r}
```

[Copy code](#)

```
library(ggplot2)
```

```
ggplot(penguins, aes(x = sex, fill = sex)) +  
  geom_bar() +  
  theme_minimal()
```

```
yaml
```

[Copy code](#)

Click **Render** → you're done.

```
---
```

```
## 6 Key differences vs R Markdown (exam gold)
```

Feature	R Markdown	Quarto
File extension	`.Rmd`	`.qmd`
Output field	`output:`	`format:`
Multi-language	Mostly R	R, Python, Julia
Rendering	`knit`	`render`
Future-proof	Legacy	Actively developed

```
---
```

```
## 7 When you should use Quarto
```