# STAT 5320 - ASSIGNMENT 3

Pham Thi Thai - T00727094

March 28, 2024

## Question 1

Because your response is binary (0 or 1), and you are modeling its expected value, what is your kernel smoother actually modeling? Hint: one of two words, both of which start with p.

**Answer**: The kernel smoother is actually modeling the **probability** of the response variable being 1 (or being a male Gentoo penguin).

## Question 2

You will create a kernel smoother using all of the other continuous measurements and evaluate each of the following two approaches (four combinations in total). Choose some simple method of correctness regarding whether your estimated kernel smoother guesses if it is a Gentoo correctly. As part of your analysis, I want to see which combination of the four seems to perform best(or whether several perform approximately the same). I also want you to explain any other decisions that you had to make in determining this. Your answer should make sense to anyone in the class who has not seen the assignment, but attended class all the way up and including to the kernel smoother class.

**Answer**:

Before proceeding to create the kernel smoother, we conducted several Exploratory Data Analyses (EDAs) to gain insights into the data set.

Firstly, we generate a scatter plot matrix for the continuous variables, grouped by species. This matrix helps visualize the relationships between variables and identify any patterns or trends. Additionally, a correlation analysis is performed to quantify the strength and direction of these relationships. Density plots are created to visualize the distribution of 'isGentoo' across the four continuous variables. These plots provide insights into the distributional characteristics of the data and highlight clear differences in distributions between Gentoo and the other two species. (refer to figure 1.)

The training errors were evaluated for four distinct approaches aimed at estimating whether a penguin is a male Gentoo based on continuous variables. These methodologies

involved constructing a kernel smoother, where the representation of continuous variables was done using both z-scores and quantiles. The kernel smoother utilized either Euclidean or Manhattan distances to compute weights for neighboring data points. These weights decayed with increasing distance from the target point, thereby localizing the estimation. The bandwidth values, dictating the width of the neighborhood, were varied from 0.1 to 1 to assess their impact on performance. Training error was computed by comparing the predicted binary outcomes with the actual labels, with 0.5 as the threshold.

- For the Z-score approach:

  With Euclidean distance, the training error remained consistently low (0%) across all bandwidth values.

  With Manhattan distance, the training error was also consistently low (0%) for all bandwidth values.

- For the Quantiles approach:

  With Euclidean distance, the training error started at 0% for a bandwidth of 0.1 and increased to 63.69% for a bandwidth of 1. With Manhattan distance, the training error started at 0% for a bandwidth of 0.1 and increased to 63.69% for a bandwidth of 1. The training errors for various bandwidth values and methods are summarized as in table 1.

  These results suggest that the Z-score method outperformed the Quantiles method in terms of training error, with consistently lower errors across different bandwidth values and distance metrics.

Table 1: Kernel Smoother Evaluation Results

| Bandwidth | Z-scored. Euclidean | Z-scored. Manhattan | Quantiles. Euclidean | Quantiles. Manhattan |
|---|---|---|---|---|
| 0.1 | 0.0000000 | 0.0000000 | 0.0000000 | 0.00000000 |
| 0.3 | 0.0000000 | 0.0000000 | 0.3095238 | 0.01190476 |
| 0.5 | 0.0000000 | 0.0000000 | 0.6369048 | 0.25000000 |
| 0.7 | 0.0000000 | 0.0000000 | 0.6369048 | 0.52976190 |
| 1.0 | 0.1964286 | 0.0000000 | 0.6369048 | 0.63690476 |

# Appendix

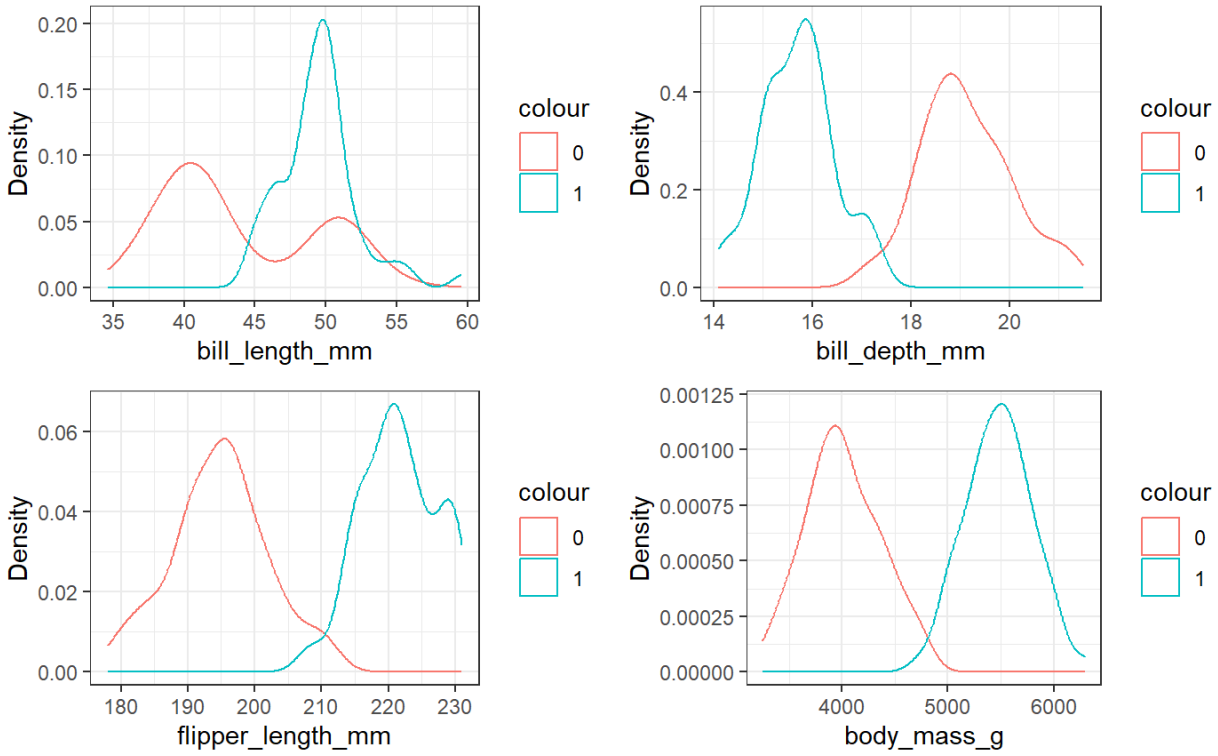


Figure 1: Density plots group by IsGentoo

## R Code

```
# KERNEL SMOOTHING METHOD
```{r}
# Load the data into the variable named 'data'
data <- palmerpenguins::penguins

# Filter the data for male penguins
maledata <- subset(data, sex == "male")

# Indicating whether the species is Gentoo (1) or not (0)
isGentoo <- ifelse(maledata$species == "Gentoo", 1, 0)

# Add the isGentoo column to maledata
maledata$isGentoo <- ifelse(maledata$species == "Gentoo", 1, 0)

# Select continuous variables for modeling
continuous_vars <- c("bill_length_mm", "bill_depth_mm", "flipper_length_mm", "body_
```


## Density plot grouped by IsGentoo
```{r}
# Load necessary libraries
library(ggplot2)

# Create density plots for each continuous variable, grouped by IsGentoo
density_plots <- lapply(names(maledata)[3:6], function(var) {
 ggplot(maledata, aes_string(x = var, color = factor(isGentoo))) +
 geom_density(alpha = 0.5) +
 labs(x = var, y = "Density") +
 theme_bw()
})

# Print density plots in a 2x2 grid
gridExtra::grid.arrange(grobs = density_plots, ncol = 2)
```


## Kernel Density Estimation (KDE)
```{r}
# Load required libraries
library(ggplot2)

# Select continuous variables for KDE
continuous_vars <- c("bill_length_mm", "bill_depth_mm", "flipper_length_mm", "body_
```
```

```
# Create KDE plots for each continuous variable
kde_plots <- lapply(continuous_vars, function(var) {
 ggplot(maledata, aes(x = !!sym(var), fill = species)) +
 geom_density(alpha = 0.5) +
 labs(title = paste("KDE Plot for", var)) +
 theme_minimal()
})

# Arrange KDE plots in a grid
gridExtra::grid.arrange(grobs = kde_plots, ncol = 2)
```

```

## Apply Kernel smoothing method using different approaches

```{r}
# Method 1: Using z-scores on each of the variables
scaled_data_zscore <- scale(maledata[continuous_vars])

# Method 2: Turning all (univariate) variables into quantiles
scaled_data_quantiles <- apply(maledata[continuous_vars], 2, function(x) ecdf(x)(x)

# Define function to calculate kernel smoother
kernel_smoother <- function(x, y, h, distance_metric) {
 n <- nrow(x)
 predictions <- rep(0, n)

 for (i in 1:n) {
  weights <- rep(0, n)

  for (j in 1:n) {
   if (distance_metric == "Euclidean") {
    dist_ij <- sqrt(sum((x[i,] - x[j,])^2))
   } else if (distance_metric == "Manhattan") {
    dist_ij <- sum(abs(x[i,] - x[j,]))
   }

   weights[j] <- exp(-dist_ij^2 / (2 * h^2))
  }

  predictions[i] <- sum(weights * y)
 }

 return(predictions)
```
```

```
}

# Define function to calculate training error
calculate_training_error <- function(predictions, actual) {
 result <- ifelse(predictions > 0.5, 1, 0) == actual
 accuracy <- sum(result) / length(result)
 training_error <- 1 - accuracy  # Training error is 1 - accuracy
 return(training_error)
}

# Define a range of bandwidth values to try
bandwidth_values <- c(0.1, 0.3, 0.5, 0.7, 1)

# Initialize variables to store results
training_errors <- matrix(NA, nrow = length(bandwidth_values), ncol = 4,
dimnames = list(bandwidth_values,
c("Z-score_Euclidean", "Z-score_Manhattan",
"Quantiles_Euclidean", "Quantiles_Manhattan")))

# Loop through each bandwidth value
for (i in 1:length(bandwidth_values)) {
 bw <- bandwidth_values[i]

 # Method 1: Z-scores with Euclidean distance
 predictions_zscore_euclidean <- kernel_smoother(scaled_data_zscore, isGentoo, bw,
 training_errors[i, "Z-score_Euclidean"] <- calculate_training_error(predictions_zs

 # Method 2: Z-scores with Manhattan distance
 predictions_zscore_manhattan <- kernel_smoother(scaled_data_zscore, isGentoo, bw,
 training_errors[i, "Z-score_Manhattan"] <- calculate_training_error(predictions_zs

 # Method 3: Quantiles with Euclidean distance
 predictions_quantiles_euclidean <- kernel_smoother(scaled_data_quantiles, isGentoo
 training_errors[i, "Quantiles_Euclidean"] <- calculate_training_error(predictions_

 # Method 4: Quantiles with Manhattan distance
 predictions_quantiles_manhattan <- kernel_smoother(scaled_data_quantiles, isGentoo
 training_errors[i, "Quantiles_Manhattan"] <- calculate_training_error(predictions_
}

# Print all results including bandwidth and training error for each approach
cat("Training Errors for Different Bandwidths and Methods:\n")
print(training_errors)

```
```
```