Case Study 2 - K Nearest Neighbors Solutions

Today's Goal

Welcome to the second case study in our series focusing on the K-Nearest Neighbors(KNN) algorithm. In this case study, we will continue our exploration of addressing the prevalent health concern of obesity through predictive modeling. Our main goal today is to leverage the KNN algorithm to classify individuals into obesity categories based on their unique characteristics.

Dataset Overview

For this case study, we'll continue working with the dataset used in our first case study on neural networks. This dataset contains extensive information on individuals' demographic characteristics, physical attributes, and lifestyle habits. To streamline our analysis, we'll utilize the tidy version of the dataset, where numerical variables have been rounded and a Body Mass Index (BMI) variable has been added.

Before we start our analysis, please download the dataset 'obesity_cleaned.csv' from the GitHub folder 'case study 2'.

The original dataset source: https://www.kaggle.com/datasets/mrsimple07/obesity-prediction/data.

1. Getting Started

A. Initial Setups

(i) To begin, make sure you have the following libraries installed and loaded in your R environment: tidyverse, ggplot2, dplyr, caret and class.

```
# install.packages('class')
# install.packages('caret')
library(tidyverse)
library(dplyr)
library(ggplot2)
library(caret)
library(class)
```

(ii) Set the seed to '123' to ensure the reproducibility of your results.

Recap: Why is it important to set a seed at the beginning of our analysis?

```
set.seed(123)
```

(iii) Now, please load the dataset that you've downloaded into your R environment. Assign this data to an object named 'knn_data'.

```
knn_data <- read.csv('obesity_cleaned.csv')
```

B. Data Cleanings

(i) Once again, familiarize yourself with the structure of the dataset.

Hint: You can use functions like `head()` and `glimpse()` to get an overview of the data.

```
head(knn_data)
```

```
##   Age Gender Height Weight PhysicalActivityLevel ObesityCategory  Race
## 1  56   Male  173.6   72.0                     4   Normal weight Asian
## 2  69   Male  164.1   90.0                     2           Obese Asian
## 3  46 Female  168.1   72.9                     4      Overweight Black
## 4  32   Male  168.5   84.9                     3      Overweight Black
## 5  60   Male  183.6   69.0                     3   Normal weight Asian
## 6  25 Female  166.4   61.1                     4   Normal weight Other
##   MaritalStatus DailyCalorieIntake SleepHours  BMI
## 1      Divorced             1849.6        8.4 23.9
## 2       Widowed             1876.1        8.4 33.4
## 3       Married             2251.9        5.9 25.8
## 4      Divorced             1642.4        8.0 29.9
## 5       Widowed             1964.2        7.7 20.5
## 6      Divorced             2051.8        6.5 22.1
```

```
# glimpse(knn_data)
```

(ii) Generate a statistical summary for the numerical variables in the dataset.

Hint: Try using the function `summary()`.

```
summary(knn_data)
```

```
##       Age           Gender              Height          Weight
##  Min.   :18.00   Length:1000        Min.   :136.1   Min.   : 26.10
##  1st Qu.:35.00   Class :character   1st Qu.:163.5   1st Qu.: 61.10
##  Median :50.00   Mode  :character   Median :169.8   Median : 71.95
##  Mean   :49.86                      Mean   :170.1   Mean   : 71.21
##  3rd Qu.:66.00                      3rd Qu.:177.3   3rd Qu.: 81.10
##  Max.   :79.00                      Max.   :201.4   Max.   :118.90
##  PhysicalActivityLevel ObesityCategory        Race           MaritalStatus
##  Min.   :1.000         Length:1000        Length:1000        Length:1000
##  1st Qu.:2.000         Class :character   Class :character   Class :character
##  Median :3.000         Mode  :character   Mode  :character   Mode  :character
##  Mean   :2.534
##  3rd Qu.:4.000
##  Max.   :4.000
##  DailyCalorieIntake   SleepHours          BMI
##  Min.   :1500       Min.   : 4.000   Min.   : 8.50
##  1st Qu.:1838       1st Qu.: 6.800   1st Qu.:20.90
##  Median :1997       Median : 7.550   Median :24.70
##  Mean   :1998       Mean   : 7.514   Mean   :24.89
##  3rd Qu.:2160       3rd Qu.: 8.300   3rd Qu.:28.70
##  Max.   :2763       Max.   :10.000   Max.   :50.80
```

(iii) What are the minimum, maximum, mean, and median values of sleeping hours for the individuals in the dataset?

- **Min**: 4.00
- **Max**: 10.00
- **Mean**: 7.51
- **Median**: 7.55

(iv) Transform all categorical variables into their factor formats. By doing so, we ensure that R recognizes

and treats these variables as categorical in the visualization the modeling tasks.

Hint: You can use the function `as.factor()`.

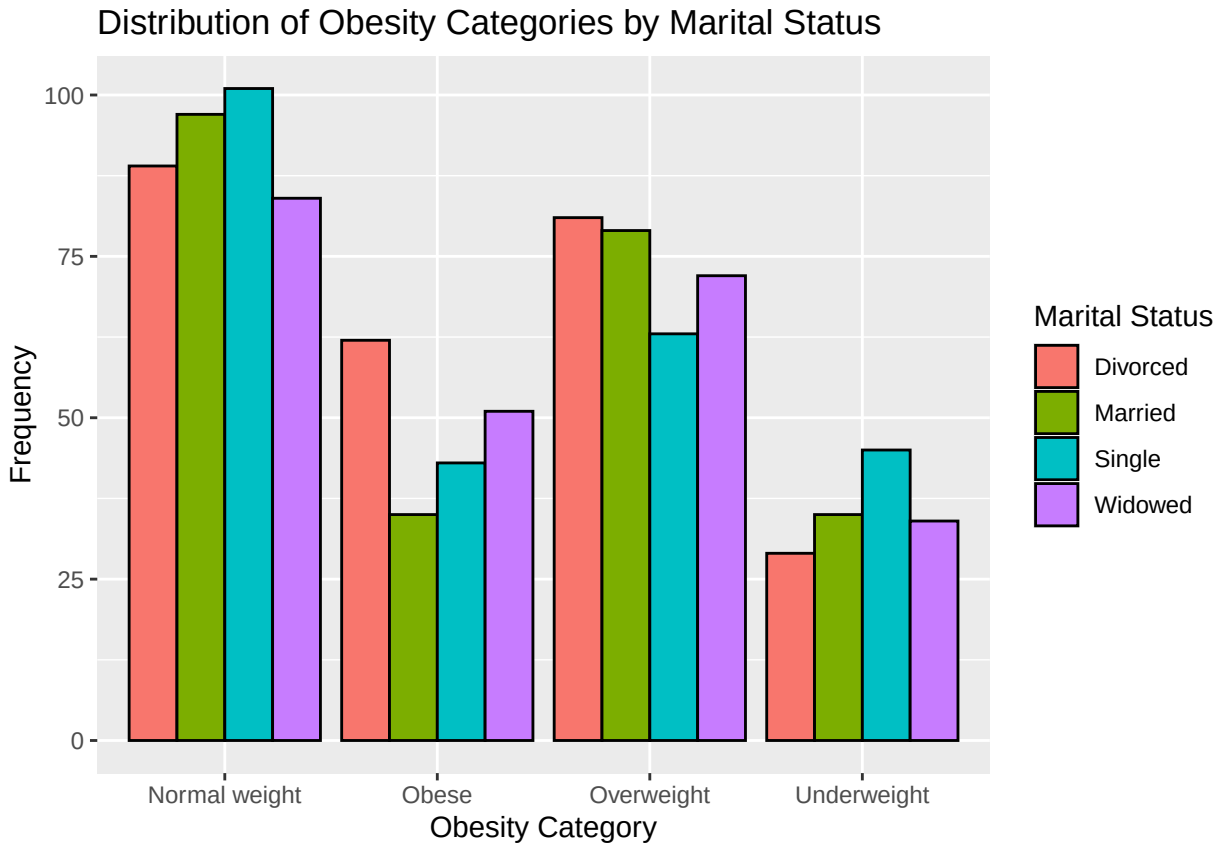
```
knn_data$Gender <- as.factor(knn_data$Gender)
knn_data$PhysicalActivityLevel <- as.factor(knn_data$PhysicalActivityLevel)
knn_data$ObesityCategory <- as.factor(knn_data$ObesityCategory)
knn_data$Race <- as.factor(knn_data$Race)
knn_data$MaritalStatus <- as.factor(knn_data$MaritalStatus)
```

2. Data Visualizations

(i) Create a bar plot to visualize the distribution of obesity categories among different marital statuses in the dataset. Make sure you generate a plot with clear title and axis label.

Hint: To differentiate between different marital statuses, use distinct colors for the bars to represent different marital status by including `fill = MaritalStatus` within the `aes()` argument.

```
ggplot(knn_data, aes(x = ObesityCategory, fill = MaritalStatus)) +
  geom_bar(position = "dodge", color = "black") +
  labs(title = "Distribution of Obesity Categories by Marital Status",
       x = "Obesity Category",
       y = "Frequency",
       fill = "Marital Status")
```



- `position = "dodge"`: This parameter specifies the positioning of the bars in the plot. When set to "dodge", it means that the bars for different levels of the MaritalStatus variable will be positioned side by side, making it easier to compare the distribution of obesity categories across different marital statuses.

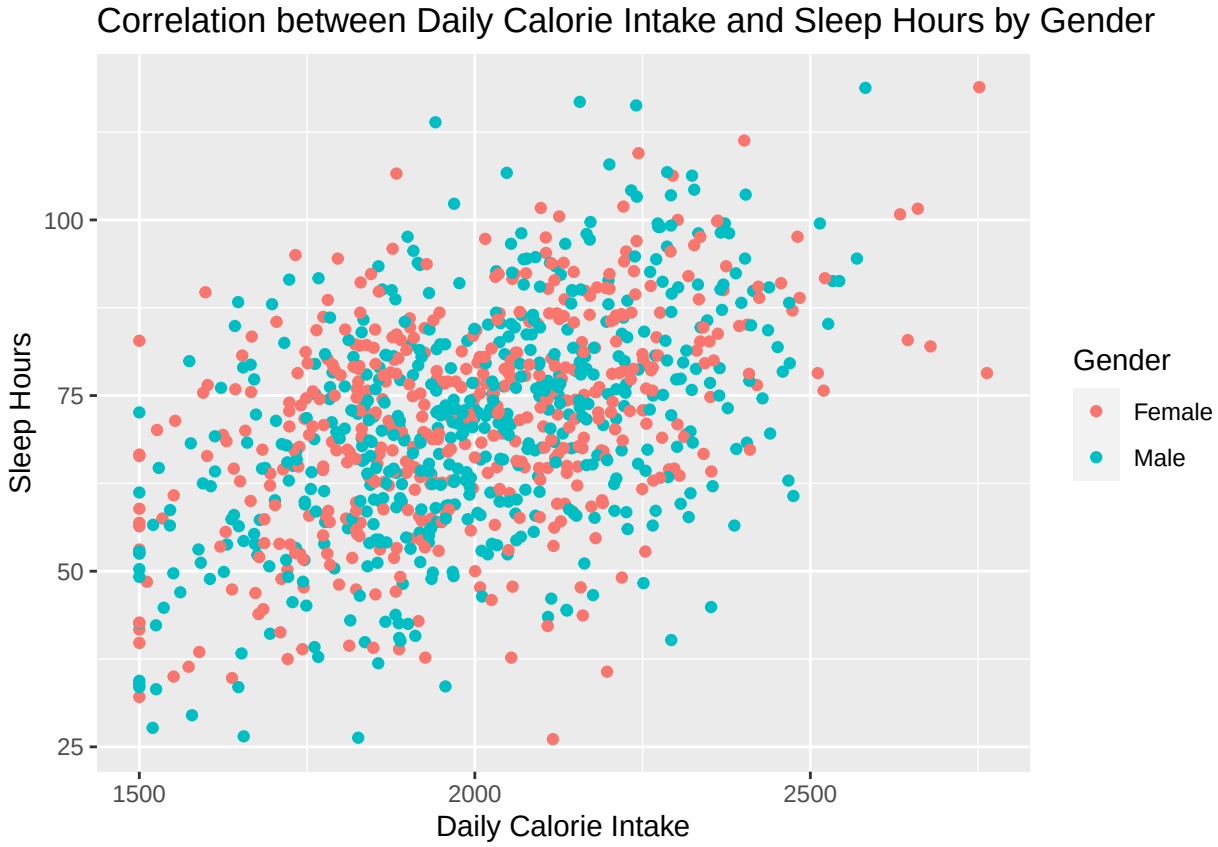
- `color = "black"`: This parameter sets the color of the outlines of the bars. By specifying `color = "black"`, the borders of the bars will be drawn in black color.
- `fill = "Marital Status"`: This parameter maps the MaritalStatus variable to the fill aesthetic of the bars. Each level of the MaritalStatus variable will be represented by a different color, allowing us to visually differentiate between the bars corresponding to different marital statuses.

(ii) Briefly describe the distributions you observed from the generated bar plot.

Based on the bar plot, the frequency of individuals with normal weight is highest among those who are single, followed by those who are married, divorced, and then widowed, with the widowed group having the lowest frequency. For the obese and overweight categories, individuals who are divorced appear to be the most represented, while in the underweight category, single individuals are the most prevalent. Overall, there are fewer individuals in the dataset who fall into the obese and underweight categories compared to the normal weight and overweight categories.

(iii) Create a scatter plot to visualize the correlation between daily calorie intake and weight, differentiated by gender. Again, make sure you generate a plot with clear title and axis label.

```
ggplot(knn_data, aes(x = DailyCalorieIntake, y = Weight, color = Gender)) +
  geom_point() +
  labs(title = "Correlation between Daily Calorie Intake and Sleep Hours by Gender",
       x = "Daily Calorie Intake",
       y = "Sleep Hours")
```



(iv) What can you infer from the scatter plot?

According to the scatter plot, there is a moderate positive correlation between daily calorie intake and weight. The plot also shows that the correlation between calorie intake and weight is similar for both genders, as evidenced by the interspersed distribution of red (female) and blue (male) points across the entire range of calorie intake values.

3. Model Building

(i) Before building our KNN models, we need to first scale all numerical variables in our dataset to ensure they all contribute equally to the analysis. Use the `scale()` function to normalize these variables.

```
variables_to_scale <- c("Age", "Height", "Weight", "BMI", "DailyCalorieIntake", "SleepHours")
knn_data[variables_to_scale] <- scale(knn_data[variables_to_scale])
```

(ii) Similar to what we've done in case study 1, convert 'Gender', 'Race', and 'MaritalStatus' into their numeric formats.

```
knn_data$Gender <- as.numeric(factor(knn_data$Gender))
knn_data$Race <- as.numeric(factor(knn_data$Race))
knn_data$MaritalStatus <- as.numeric(factor(knn_data$MaritalStatus))
```

(iii) Divide the dataset into 70% training and 30% testing sets.

```
data_rows <- floor(0.7 * nrow(knn_data))
train_indices <- sample(c(1:nrow(knn_data)), data_rows)
train_data <- knn_data[train_indices,]
test_data <- knn_data[-train_indices,]
```

(iv) Remove the response variable 'ObesityCategory' from the training and testing datasets, and store the resulting datasets into new objects titled 'train_scaled' and 'test_scaled'.

Note: Don't change the original traing data 'train_data' and testing data 'test_data'.

```
train_scaled <- train_data %>% select(-ObesityCategory)
test_scaled <- test_data %>% select(-ObesityCategory)
```

Discussion: Why do we need to remove the respones variable before we apply the KNN algorithm to make predictions on our dataset?

KNN uses the features (independent variables) of the k nearest neighbors to classify a new observation. Including the response variable in the feature set would give the model direct access to the answer, which could then leads to overfitting.

(v) Constructing the KNN Model

- Use the `knn()` function to build the KNN model and make predictions. Recall that our goal is to classify individuals into specific obesity categories.
- Use all available covariates in the dataset.
- Set the number of neighbors k to 1.
- Store the predictions in a variable named test_pred.

```
test_pred <- knn(
                 train = train_scaled,
                 test = test_scaled,
                 cl = train_data$ObesityCategory,
                 k=1
                 )
```

(vi) Evaluate the model's performance by generating the confusion matrix for the KNN predictions using the function confusionMatrix().

```
confusionMatrix <- confusionMatrix(as.factor(test_pred), as.factor(test_data$ObesityCategory))
print(confusionMatrix)
```

```
## Confusion Matrix and Statistics
##
##                Reference
## Prediction      Normal weight Obese Overweight Underweight
##   Normal weight            73     0         19          13
##   Obese                     0    40         15           0
##   Overweight               25    16         55           0
##   Underweight              10     0          0          34
##
## Overall Statistics
##
##                Accuracy : 0.6733
##                  95% CI : (0.6171, 0.7261)
##     No Information Rate : 0.36
##     P-Value [Acc > NIR] : < 2.2e-16
##
##                   Kappa : 0.5475
##
##  Mcnemar's Test P-Value : NA
##
## Statistics by Class:
##
##                      Class: Normal weight Class: Obese Class: Overweight
## Sensitivity                        0.6759       0.7143            0.6180
## Specificity                        0.8333       0.9385            0.8057
## Pos Pred Value                     0.6952       0.7273            0.5729
## Neg Pred Value                     0.8205       0.9347            0.8333
## Prevalence                         0.3600       0.1867            0.2967
## Detection Rate                     0.2433       0.1333            0.1833
## Detection Prevalence               0.3500       0.1833            0.3200
## Balanced Accuracy                  0.7546       0.8264            0.7118
##                      Class: Underweight
## Sensitivity                      0.7234
## Specificity                      0.9605
## Pos Pred Value                   0.7727
## Neg Pred Value                   0.9492
## Prevalence                       0.1567
## Detection Rate                   0.1133
## Detection Prevalence             0.1467
## Balanced Accuracy                0.8419
```

(vii) What do the accuracy, sensitivity, and specificity rates indicate about the predictive power of the model?

The model exhibits moderate accuracy at 67.33%, indicating that it correctly predicts the obesity category for approximately two-thirds of the cases. The sensitivity rates reveal that the model is best at correctly identifying underweight individuals (72.34%), followed by those with obesity (71.43%), normal weight (67.59%), and overweight (61.80%). In terms of specificity, the model performs exceptionally well in distinguishing individuals who are not underweight (96.05%) and not obese (93.85%), but it is less effective in distinguishing those who are not overweight or not of normal weight, with specificity rates of 80.57% and 83.33% respectively.

These metrics suggest the model is more reliable in identifying true positives in the obese and underweight categories and true negatives in the underweight and obese categories.

(viii) Use the KNN algorithm to make predictions by setting the number of nearest neighbors to 5, 10, 20, and 100. For each model, generate the corresponding confusion matrices.

```
test_pred2 <- knn(
                  train = train_scaled,
                  test = test_scaled,
                  cl = train_data$ObesityCategory,
                  k=5
                  )
```

```
confusionMatrix2 <- confusionMatrix(as.factor(test_pred2), as.factor(test_data$ObesityCategory))
print(confusionMatrix2)
```

```
## Confusion Matrix and Statistics
##
##                Reference
## Prediction      Normal weight Obese Overweight Underweight
##   Normal weight            85     0         12          14
##   Obese                     0    46          8           0
##   Overweight               19    10         69           0
##   Underweight               4     0          0          33
##
## Overall Statistics
##
##                Accuracy : 0.7767
##                  95% CI : (0.7253, 0.8225)
##     No Information Rate : 0.36
##     P-Value [Acc > NIR] : < 2.2e-16
##
##                   Kappa : 0.6885
##
##  Mcnemar's Test P-Value : NA
##
## Statistics by Class:
##
##                      Class: Normal weight Class: Obese Class: Overweight
## Sensitivity                        0.7870       0.8214            0.7753
## Specificity                        0.8646       0.9672            0.8626
## Pos Pred Value                     0.7658       0.8519            0.7041
## Neg Pred Value                     0.8783       0.9593            0.9010
## Prevalence                         0.3600       0.1867            0.2967
## Detection Rate                     0.2833       0.1533            0.2300
## Detection Prevalence               0.3700       0.1800            0.3267
## Balanced Accuracy                  0.8258       0.8943            0.8189
##                      Class: Underweight
## Sensitivity                      0.7021
## Specificity                      0.9842
## Pos Pred Value                   0.8919
## Neg Pred Value                   0.9468
## Prevalence                       0.1567
## Detection Rate                   0.1100
## Detection Prevalence             0.1233
```

```
## Balanced Accuracy                0.8432
```

```
test_pred3 <- knn(
                train = train_scaled,
                test = test_scaled,
                cl = train_data$ObesityCategory,
                k=10
                )
```

```
confusionMatrix3 <- confusionMatrix(as.factor(test_pred3), as.factor(test_data$ObesityCategory))
print(confusionMatrix3)
```

```
## Confusion Matrix and Statistics
##
##                Reference
## Prediction      Normal weight Obese Overweight Underweight
##   Normal weight            88     0         12          22
##   Obese                     0    35          5           0
##   Overweight               18    21         72           0
##   Underweight               2     0          0          25
##
## Overall Statistics
##
##                Accuracy : 0.7333
##                  95% CI : (0.6795, 0.7825)
##     No Information Rate : 0.36
##     P-Value [Acc > NIR] : < 2.2e-16
##
##                   Kappa : 0.6217
##
##  Mcnemar's Test P-Value : NA
##
## Statistics by Class:
##
##                      Class: Normal weight Class: Obese Class: Overweight
## Sensitivity                        0.8148       0.6250            0.8090
## Specificity                        0.8229       0.9795            0.8152
## Pos Pred Value                     0.7213       0.8750            0.6486
## Neg Pred Value                     0.8876       0.9192            0.9101
## Prevalence                         0.3600       0.1867            0.2967
## Detection Rate                     0.2933       0.1167            0.2400
## Detection Prevalence               0.4067       0.1333            0.3700
## Balanced Accuracy                  0.8189       0.8023            0.8121
##                      Class: Underweight
## Sensitivity                     0.53191
## Specificity                     0.99209
## Pos Pred Value                  0.92593
## Neg Pred Value                  0.91941
## Prevalence                      0.15667
## Detection Rate                  0.08333
## Detection Prevalence            0.09000
## Balanced Accuracy               0.76200
```

```
test_pred4 <- knn(
                train = train_scaled,
```

```
                test = test_scaled,
                cl = train_data$ObesityCategory,
                k=20
                )
```

```
confusionMatrix4 <- confusionMatrix(as.factor(test_pred4), as.factor(test_data$ObesityCategory))
print(confusionMatrix4)
```

```
## Confusion Matrix and Statistics
##
##                Reference
## Prediction      Normal weight Obese Overweight Underweight
##   Normal weight            96     0         19          22
##   Obese                     0    36          4           0
##   Overweight               12    20         66           0
##   Underweight               0     0          0          25
##
## Overall Statistics
##
##                Accuracy : 0.7433
##                  95% CI : (0.69, 0.7918)
##     No Information Rate : 0.36
##     P-Value [Acc > NIR] : < 2.2e-16
##
##                   Kappa : 0.6337
##
##  Mcnemar's Test P-Value : NA
##
## Statistics by Class:
##
##                      Class: Normal weight Class: Obese Class: Overweight
## Sensitivity                        0.8889       0.6429            0.7416
## Specificity                        0.7865       0.9836            0.8483
## Pos Pred Value                     0.7007       0.9000            0.6735
## Neg Pred Value                     0.9264       0.9231            0.8861
## Prevalence                         0.3600       0.1867            0.2967
## Detection Rate                     0.3200       0.1200            0.2200
## Detection Prevalence               0.4567       0.1333            0.3267
## Balanced Accuracy                  0.8377       0.8132            0.7950
##                      Class: Underweight
## Sensitivity                     0.53191
## Specificity                     1.00000
## Pos Pred Value                  1.00000
## Neg Pred Value                  0.92000
## Prevalence                      0.15667
## Detection Rate                  0.08333
## Detection Prevalence            0.08333
## Balanced Accuracy               0.76596
```

```
test_pred5 <- knn(
                train = train_scaled,
                test = test_scaled,
                cl = train_data$ObesityCategory,
                k=100
```

```
                )
```

```
confusionMatrix5 <- confusionMatrix(as.factor(test_pred5), as.factor(test_data$ObesityCategory))
print(confusionMatrix5)
```

```
## Confusion Matrix and Statistics
##
##                Reference
## Prediction      Normal weight Obese Overweight Underweight
##   Normal weight           100     0         26          39
##   Obese                     0    26          2           0
##   Overweight                8    30         61           0
##   Underweight               0     0          0           8
##
## Overall Statistics
##
##                Accuracy : 0.65
##                  95% CI : (0.5931, 0.7039)
##     No Information Rate : 0.36
##     P-Value [Acc > NIR] : < 2.2e-16
##
##                   Kappa : 0.4872
##
##  Mcnemar's Test P-Value : NA
##
## Statistics by Class:
##
##                      Class: Normal weight Class: Obese Class: Overweight
## Sensitivity                        0.9259      0.46429            0.6854
## Specificity                        0.6615      0.99180            0.8199
## Pos Pred Value                     0.6061      0.92857            0.6162
## Neg Pred Value                     0.9407      0.88971            0.8607
## Prevalence                         0.3600      0.18667            0.2967
## Detection Rate                     0.3333      0.08667            0.2033
## Detection Prevalence               0.5500      0.09333            0.3300
## Balanced Accuracy                  0.7937      0.72804            0.7526
##                      Class: Underweight
## Sensitivity                     0.17021
## Specificity                     1.00000
## Pos Pred Value                  1.00000
## Neg Pred Value                  0.86644
## Prevalence                      0.15667
## Detection Rate                  0.02667
## Detection Prevalence            0.02667
## Balanced Accuracy               0.58511
```

(ix) Based on the performance metrics obtained for different values of 'k' in the KNN algorithm, what can you conclude about the impact of 'k' on the model's predictive accuracy?

As per the performance metrics, increasing 'k' from 1 to 20 leads to an improvement in classification accuracy, with the peak accuracy of 77.67% observed at k=5. This indicates that a moderate number of neighbors, particularly around 5 for our task, optimizes the balance between capturing the underlying patterns in the data and avoiding overfitting. To conclude, a very small 'k' might lead to overfitting by being too sensitive to noise in the data, whereas a very large 'k' tends to overly generalize, thereby missing critical data nuances, which is reflected in the reduced accuracy when k=100.

(x) Discuss the advantages and disadvantages of applying KNN versus Neural Networks for our classification tasks.

Advantages of KNN

- Since KNN is a lazy learning algorithm, it doesn't require training a model. This can save time and computational resources, especially for smaller datasets. Neural Networks require a significant amount of data and computational resources for training.
- KNN is straightforward to understand and implement. As comparisons, Neural Networks are often considered "black boxes" due to their complex structure, making it difficult to understand how decisions are made.

Disadvantages of KNN

- KNN's performance deteriorates as the dataset size increases, due to the need to compute the distance between the test sample and each training example, making it less efficient for large datasets.
- Since distance measurements are at the core of KNN, the presence of irrelevant or redundant features can significantly impact the algorithm's performance, requiring careful feature selection.
- KNN stores all the training data, which can lead to high memory usage for large datasets, making it impractical for applications with memory constraints.