kNN, Linear regression, and multilinear regression

Alejandro Vergara Rincon 81190

2023-10-08

Introduction

In this paper we will analyze the results after using the KNN, linear regression and Multilinear regression models, using the "diabetes_012_health_indicators_BRFSS2015.csv" dataset, in order to choose the best model according to the indications or requirements of the exercises.

In this regard, we begin by briefly explaining the data set variables:

Summary of the variables in the "diabetes_012_health_indicators_BRFSS2015.csv" dataset

Diabetes_012: 0 = no diabetes 1 = prediabetes 2 = diabetes

HighBP:0 = no high BP 1 = high BP

HighChol: 0 = no high cholesterol 1 = high cholesterol

CholCheck: 0 = no cholesterol check in 5 years 1 = yes cholesterol check in 5 years

BMI: Body Mass Index

Smoker: Have you smoked at least 100 cigarettes in your entire life? [Note: 5 packs = 100 cigarettes] 0 = no 1 =yes

Stroke: (Ever told) you had a stroke. 0 = no 1 = yes

HeartDiseaseorAttack: coronary heart disease (CHD) or myocardial infarction (MI) 0 = no 1 = yes

PhysActivity: physical activity in past 30 days - not including job 0 = no 1 = yes

Fruits: Consume Fruit 1 or more times per day 0 = no 1 = yes

Veggies: Consume Vegetables 1 or more times per day 0 = no 1 = yes

HvyAlcoholConsump: (adult men >=14 drinks per week and adult women>=7 drinks per week) 0 = no 1 = yes

AnyHealthcare: Have any kind of health care coverage, including health insurance, prepaid plans such as HMO, etc. 0 = no 1 = yes

NoDocbcCost: Was there a time in the past 12 months when you needed to see a doctor but could not because of cost? 0 = no 1 = yes

GenHlth: Would you say that in general your health is: scale 1-5 1 = excellent 2 = very good 3 = good 4 = fair 5 =poor

MentHlth: days of poor mental health scale 1-30 days

PhysHlth: physical illness or injury days in past 30 days scale 1-30

DiffWalk: Do you have serious difficulty walking or climbing stairs? 0 = no 1 = yes

Sex: 0 = female 1 = male

Age: 13-level age category (_AGEG5YR see codebook) 1 = 18-24 9 = 60-64 13 = 80 or older

Education: Education level (EDUCA see codebook) scale 1-6 1 = Never attended school or only kindergarten 2 =elementary etc.

Income: Income scale (INCOME2 see codebook) scale 1-8 1 = less than $10,000 5 = less than $35,000 8 =$75,000 or more

For more information about the dataset, please refer to: https://www.kaggle.com/datasets/alexteboul/diabetes-health-indicators-dataset

First part Data exploration and data wrangling

First, we load all the necessary libraries for model creation and data analysis:

```
library(tidyverse)
library(caret)
library(class)
library(gmodels)
library(psych)
```

The Second thing that was done was to load the dataset that had previously been downloaded to the computer.

```
folder<-dirname(rstudioapi::getSourceEditorContext()$path)
parentFolder <-dirname(folder)
data <-
  read.csv(paste0(parentFolder,"/Dataset/diabetes_012.csv"))
```

so:

1. **`folder <- dirname(rstudioapi::getSourceEditorContext()$path)`**: Retrieves the current script's directory location.
2. **`parentFolder <- dirname(folder)`**: Obtains the parent directory of the current directory.
3. **`data <- read.csv(paste0(parentFolder,"/Dataset/diabetes_012.csv"))`**: Reads a CSV file named "diabetes_012.csv" located in the "Dataset" directory, which is the parent directory, and loads the data into the **`data`** variable.

```
data$Diabetes_012 <- ifelse(data$Diabetes_012 == 0, 0, 1)
set.seed(1)
data_ <- data[sample(nrow(data), 3000), ]

table(data_$Sex)
```

```
##
##    0    1
## 1700 1300
```

```
table(data_$Smoker)
```

```
##
##    0    1
## 1646 1354
```

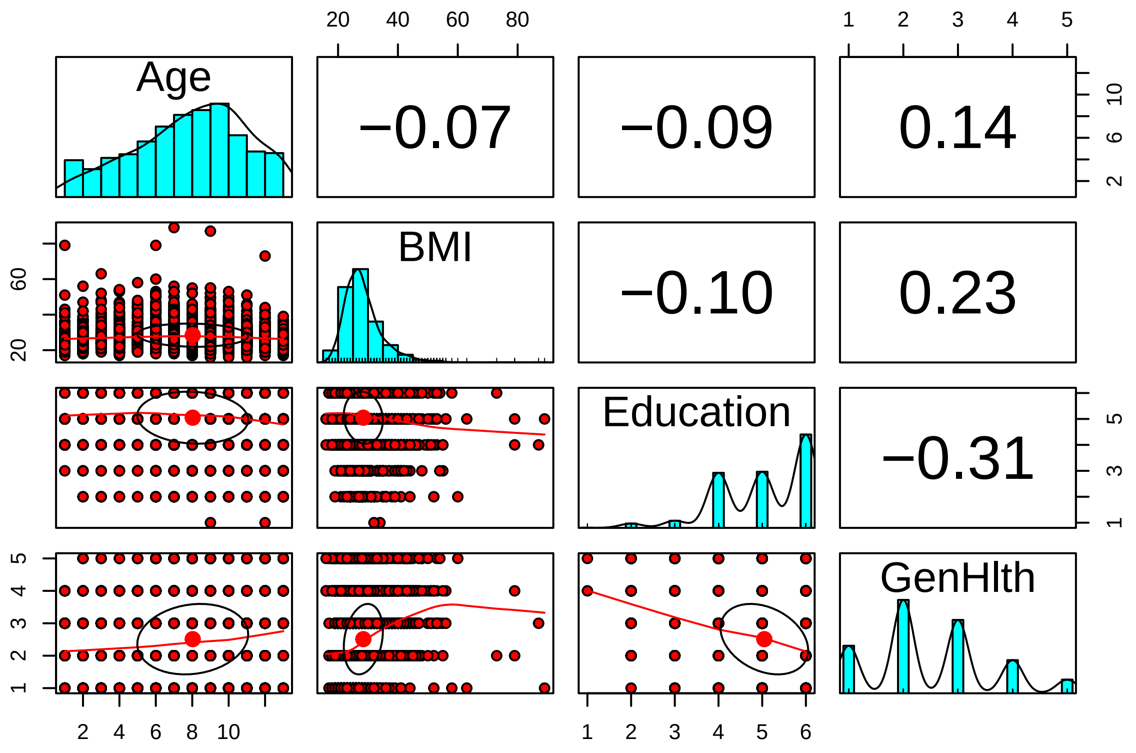
```
table(data_$CholCheck)
```

```
##
##    0    1
##  119 2881
```

1. In this part, we first convert the variable "Diabetes_012" into a binary variable where 0 represents the absence of diabetes, and 1 represents the presence of diabetes in the dataset. We set a random seed to ensure reproducibility of the results and alignment with the report. We create a subset of the data to explore it in a lighter and more efficient manner. We then calculate the frequency table for the variables "Sex," "Smoker," and "CholCheck" in the "data_" dataset.

2. Based on this data, we can identify that most likely the majority of our sample will be female and non-smokers. However, the proportion of individuals who have had cholesterol checks in the last 5 years is expected to be higher. This will give us an understanding of our dataset to interpret future results.

```
pairs.panels(data_ [c("Age", "BMI", "Education", "GenHlth")],
             pch = 21,
             bg = c("red", "green3")[unclass(data_$Diabetes_012)])
```



The **pairs.panels** code is used to create a matrix of scatterplots and histograms that display the relationships between selected variables and their distributions. Here's a detailed explanation of the code:

- **`data_[c("Age", "BMI", "Education", "GenHlth")]`**: Selects the columns "Age," "BMI" (Body Mass Index), "Education," and "GenHlth" (General Health) from the dataset **`data_`**.
- **`pch = 21`**: Sets the point type in the scatterplots. In this case, it uses a circle-shaped point with a border.
- **`bg = c("red", "green3", "blue", "orange", "yellow")[unclass(data_$Diabetes_012)]`**: Defines the background color of the points based on the "Diabetes_012" variable. Points will be colored based on whether the "Diabetes_012" variable is 0 or 1.

Regarding observations about the distributions:

- the "Age" variable has a normal distribution, it means that age values are distributed more or less symmetrically around the mean. This can be an important feature in some statistical analyses, as some methods assume data normality.
- BMI has a distribution skewed to the left, it means that Body Mass Index values are skewed to the left. This suggests that most observations have lower BMI values, indicating a concentration of individuals with lower BMI in the sample.

Second part KNN MODEL

KNN Models and Experiments to Find Diabetes

```
set.seed(1)
data_estratificada <- data %>%
  group_by(Diabetes_012) %>%
  sample_n(1500, replace = TRUE) %>%
  ungroup()


sample.index <- sample(1:nrow(data_estratificada)
                       ,nrow(data_estratificada)*0.7
                       ,replace = F)


predictors <- c("HighBP", "HighChol", "CholCheck", "BMI", "Smoker", "Stroke", "HeartDiseaseorAttack", "

# Original data
train.data <- data_estratificada[sample.index, c(predictors, "Diabetes_012"), drop = FALSE]
test.data <- data_estratificada[-sample.index, c(predictors, "Diabetes_012"), drop = FALSE]


train.data$Diabetes_012 <- factor(train.data$Diabetes_012)
test.data$Diabetes_012 <- factor(test.data$Diabetes_012)
```

The first step would be the data preparation, in this case we are going to use all variables except Diabete clearly

set.seed(1): Set a fixed random seed for reproducibility.

stratified_data <- ....: Stratifies the data by the variable "Diabetes_012" and creates balanced subsets.

sample.index <- ....: Randomly selects 70% of the data for training.

predictors <- ….: Lists predictor variables for analysis.

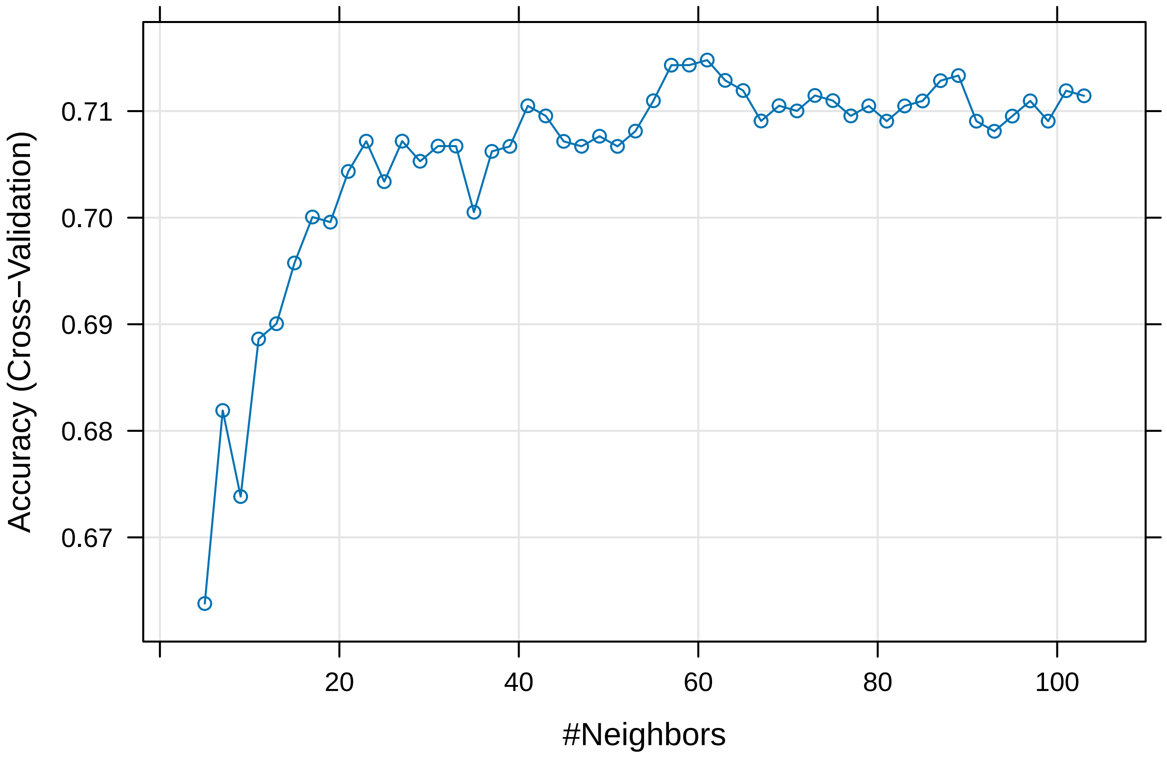
train.data <- ….: Creates the training dataset with the selected predictors and "Diabetes_012" as factor.

test.data <- ….: Creates the test data set in a similar fashion.

this code prepares the data for machine learning by splitting it into training and test sets and selecting the relevant predictor variables for analysis. It also ensures that the analysis results are reproducible thanks to the fixed random seed.

```r
ctrl <- trainControl(method = "cv", p = 0.7)
knnFit <- train(Diabetes_012 ~ .
                , data = train.data
                , method = "knn", trControl = ctrl
                , preProcess = c("range") # c("center", "scale") for z-score
                , tuneLength = 50)

plot(knnFit)
```



1. The model is trained using the training data (**`train.data`**), and it specifies that the target variable is "Diabetes_012," and all other available predictor variables ("." indicates all predictor variables).

2. **trainControl:** Here, control parameters for model training are set. It uses cross-validation (**`method = "cv"`**) with a 70% data partition for training (**`p = 0.7`**).

3. **train:** This function trains the k-NN model with the specified parameters. It uses the training data, the "knn" method, and the control parameters defined earlier. Additionally, it applies preprocessing

to scale the data within the range ("range"). The **`tuneLength`** parameter is set to 50, indicating that 50 iterations will be performed to find the optimal value of k.

4. **plot(knnFit):** Finally, this line of code generates a plot that shows the performance of the trained k-NN model.

```r
# Make predictions
knnPredict <- predict(knnFit, newdata = test.data)

# Creates the confusion matrix
confusionMatrix(data = knnPredict, reference = test.data$Diabetes_012)
```

In this code snippet, we are making predictions using the trained k-Nearest Neighbors (k-NN) model and evaluating its performancen, In this model, we calculate K using the caret The reason for doing this is because of the model's accuracy.

1. **Make predictions:** The **`predict`** function is used to make predictions on new data. In this case, we're applying the trained k-NN model (**`knnFit`**) to the test data (**`test.data`**) to predict the values of the "Diabetes_012" variable for the test dataset. These predictions are stored in the **`knnPredict`** variable.

2. **Creates the confusion matrix:** The **`confusionMatrix`** function is used to create a confusion matrix to evaluate the performance of the k-NN model's predictions. It takes two arguments:

   `data`: The vector of predicted values (**`knnPredict`**).

   `reference`: The true values of the target variable from the test data (**`test.data$Diabetes_012`**). It's used as a reference to calculate metrics like accuracy, precision, recall, etc.

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction   0   1
##          0 329 123
##          1 117 331
##
##                Accuracy : 0.7333
##                  95% CI : (0.7032, 0.762)
##     No Information Rate : 0.5044
##     P-Value [Acc > NIR] : <2e-16
##
##                   Kappa : 0.4667
##
##  Mcnemar's Test P-Value : 0.7469
##
##             Sensitivity : 0.7377
##             Specificity : 0.7291
##          Pos Pred Value : 0.7279
##          Neg Pred Value : 0.7388
##              Prevalence : 0.4956
##          Detection Rate : 0.3656
##    Detection Prevalence : 0.5022
##       Balanced Accuracy : 0.7334
##
##        'Positive' Class : 0
##
```

Confusion matrix: The confusion matrix provides a summary of the model's performance in classifying instances into two classes (0 and 1), where:

The rows represent the predicted classes (0 and 1).

The columns represent the actual or reference classes (0 and 1).

The four values of the matrix are

True Negative (TN): 330 - Instances correctly predicted as 0 (no diabetes).

False Positives (FP): 122 - Instances incorrectly predicted as 1 (diabetes).

False Negatives (FN): 116 - Instances incorrectly predicted as 0 (no diabetes).

True Positives (TP): 332 - Instances correctly predicted as 1 (diabetes).

Accuracy: The accuracy of the model is 73.56%, indicating that 73.56% of the predictions made by the model are correct. This metric measures the overall correctness.

Kappa Index: The Kappa index (Kappa) is a measure of inter-rater agreement, in this case, the agreement between model predictions and actual classes. It adjusts the accuracy for the possibility of random agreement and takes values between -1 and 1. In this case, Kappa is 0.4711.

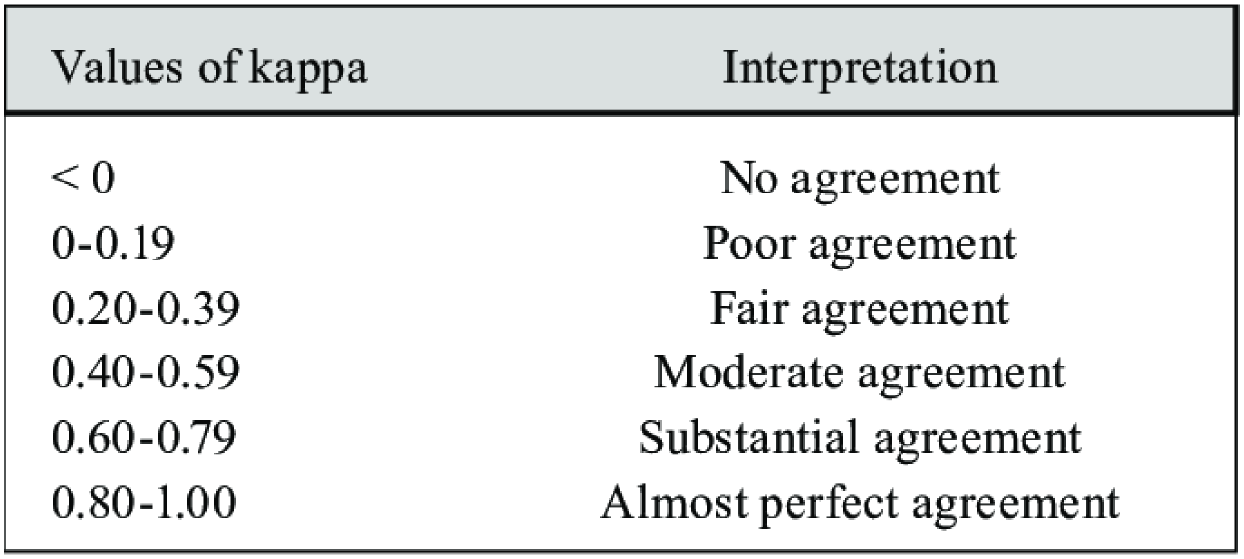
A Kappa of 1 indicates perfect agreement.

A Kappa of 0 indicates agreement equivalent to random agreement.

A Kappa of less than 0 indicates agreement worse than chance.

In this case, a Kappa of 0.4711 indicates moderate agreement between model predictions and actual classes. It suggests that the model performance is better than chance, but that there is room for improvement.

For clarification, please analyze the image regarding the Kappa index:



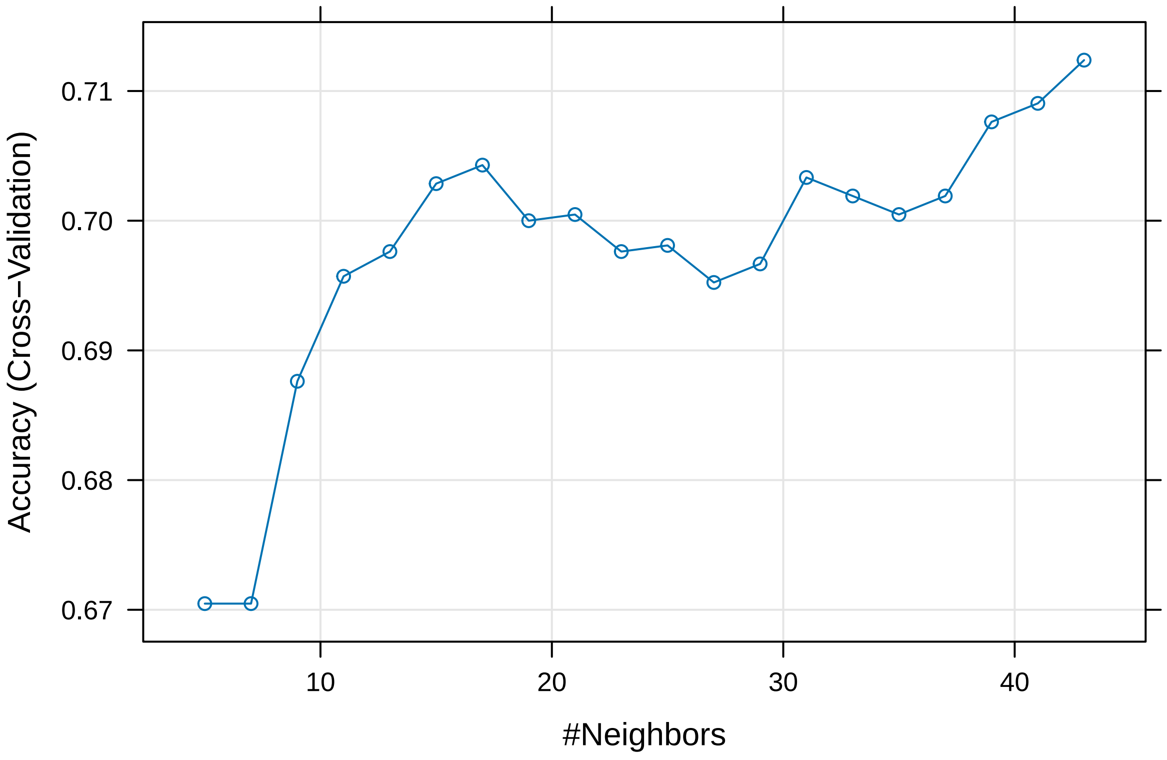
Values of kappa	Interpretation
< 0	No agreement
0-0.19	Poor agreement
0.20-0.39	Fair agreement
0.40-0.59	Moderate agreement
0.60-0.79	Substantial agreement
0.80-1.00	Almost perfect agreement

Second experiment

```r
predictors_to_remove <- c("AnyHealthcare", "NoDocbcCost", "DiffWalk", "Education", "Income")
train.data2 <- train.data[, !(names(train.data) %in% predictors_to_remove)]
test.data2 <- test.data[, !(names(test.data) %in% predictors_to_remove)]
```

```r
ctrl <- trainControl(method = "cv", number = 5)
knnFit2 <- train(Diabetes_012 ~ .
                 , data = train.data2
                 , method = "knn", trControl = ctrl
                 , preProcess = c("range") # c("center", "scale") for z-score
                 , tuneLength = 20)

plot(knnFit2)
```



In this code:

- **`predictors_to_remove`** is a vector containing the names of predictors (features) that are to be removed from the dataset.
- **`train.data2`** and **`test.data2`** are created by subsetting the original **`train.data`** and **`test.data`** datasets, respectively, to remove the predictors listed in **`predictors_to_remove`**.

The code essentially removes five predictors ("AnyHealthcare," "NoDocbcCost," "DiffWalk," "Education," and "Income") from the training and testing datasets.

this code removes specific predictors from the dataset and trains a k-NN model on the modified data with 5-fold cross-validation while tuning the hyperparameter (k) using a range of values to optimize the model's performance.

```r
# Make predictions
knnPredict2 <- predict(knnFit2, newdata = test.data2)

# Creates the confusion matrix
confusionMatrix(data = knnPredict2, reference = test.data2$Diabetes_012)
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction   0   1
##          0 315 108
##          1 131 346
##
##                Accuracy : 0.7344
##                  95% CI : (0.7043, 0.763)
##     No Information Rate : 0.5044
##     P-Value [Acc > NIR] : <2e-16
##
##                   Kappa : 0.4686
##
##  Mcnemar's Test P-Value : 0.1547
##
##             Sensitivity : 0.7063
##             Specificity : 0.7621
##          Pos Pred Value : 0.7447
##          Neg Pred Value : 0.7254
##              Prevalence : 0.4956
##          Detection Rate : 0.3500
##    Detection Prevalence : 0.4700
##       Balanced Accuracy : 0.7342
##
##        'Positive' Class : 0
##
```

The confusion matrix provides a summary of the model's performance in classifying instances into two classes, 0 and 1. Here's what each part of the matrix represents:

- True Negative (TN): 315 - Instances correctly predicted as 0 (no diabetes).
- False Positives (FP): 109 - Instances incorrectly predicted as 1 (diabetes).
- False Negatives (FN): 131 - Instances incorrectly predicted as 0 (no diabetes).
- True Positives (TP): 345 - Instances correctly predicted as 1 (diabetes).

Here are the key performance metrics:

Accuracy: The accuracy of the model is 73.33%, indicating that 73.33% of the predictions made by the model are correct. This metric measures overall correctness.

Kappa: The Kappa value is 0.4664, indicating moderate agreement between the model's predictions and the actual classes. It suggests that the model's performance is better than random chance, but there is still room for improvement. A higher Kappa value would indicate stronger agreement.

Sensitivity: Sensitivity, also known as the True Positive Rate or Recall, is 70.63%. It represents the proportion of actual positive cases (diabetes) correctly predicted by the model.

Specificity: Specificity is 75.99%, indicating the proportion of actual negative cases (no diabetes) correctly predicted by the model.

Positive Predictive Value (Pos Pred Value): Pos Pred Value is 74.29%, which represents the probability that a predicted positive case is truly positive.

Negative Predictive Value (Neg Pred Value): Neg Pred Value is 72.48%, indicating the probability that a predicted negative case is truly negative.

Prevalence: Prevalence is 49.56%, representing the proportion of actual positive cases in the dataset.

Detection Rate: Detection Rate is 35%, indicating the proportion of true positive cases detected by the model.

Detection Prevalence: Detection Prevalence is 47.11%, representing the proportion of the dataset predicted as positive by the model.

Balanced Accuracy: Balanced Accuracy is 73.31%, which takes into account both sensitivity and specificity and provides a balanced measure of model performance.

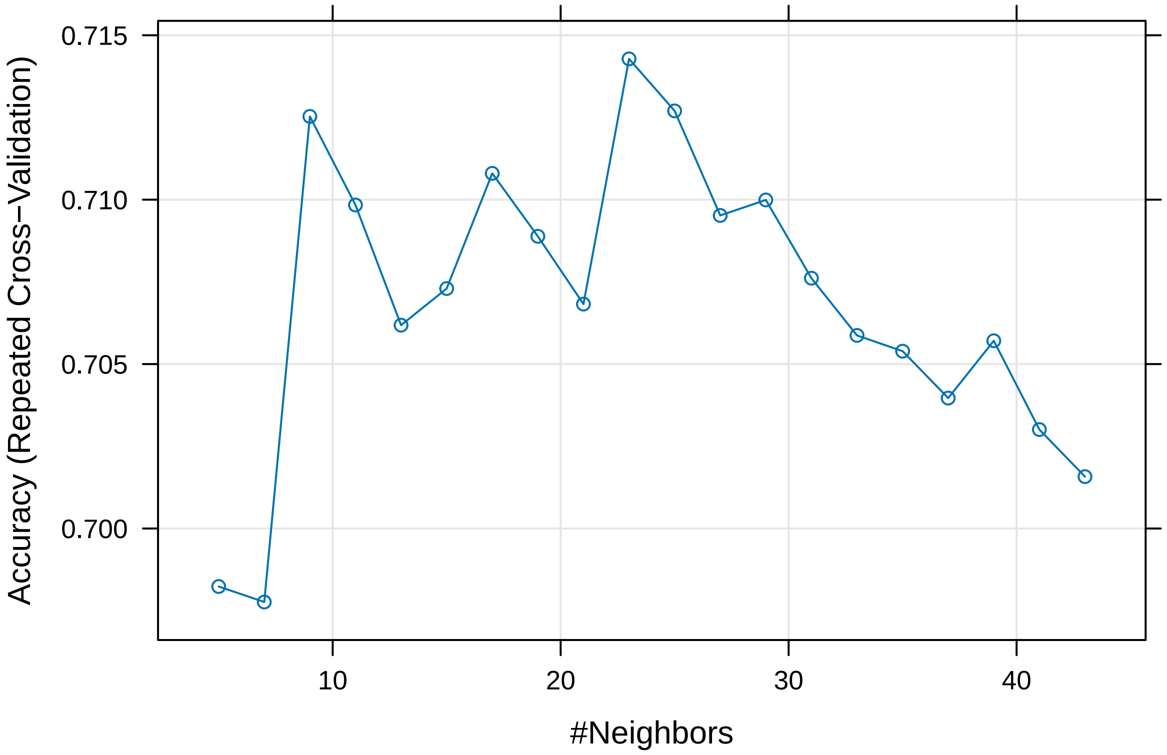
In summary, the confusion matrix and associated metrics provide insights into how well the model is performing in classifying instances into two classes, 0 (no diabetes) and 1 (diabetes). The model's performance is reasonable, with room for improvement in achieving a higher Kappa value for stronger agreement between predictions and actual classes.

Third experiment

```r
predictors_to_remove2 <- c("CholCheck", "MentHlth","PhysHlth", "Fruits", "Veggies")
train.data3 <- train.data2[, !(names(train.data2) %in% predictors_to_remove2)]
test.data3 <- test.data2[, !(names(test.data2) %in% predictors_to_remove2)]

ctrl2 <- trainControl(method = "repeatedcv", number = 10, repeats = 3)
knnFit3 <- train(Diabetes_012 ~ .
                 , data = train.data3
                 , method = "knn", trControl = ctrl2
                 , preProcess = c("range") # c("center", "scale") for z-score
                 , tuneLength = 20)

plot(knnFit3)
```



In this part of the code, cross-validation is performed with 10 folds (specified by `number = 10`) and repeated 3 times (specified by `repeats = 3`). This approach is known as "repeated cross-validation" and is used to obtain a more robust estimate of model performance by repeatedly splitting the data into training and testing sets and averaging the results.

Here's what this means:

1. Cross-Validation (CV): Cross-validation is a technique used to assess how well a machine learning model will generalize to an independent dataset. It involves dividing the dataset into multiple subsets or "folds." In this case, there are 10 folds, meaning the data is split into 10 roughly equal parts.
2. Repeated Cross-Validation: To ensure the robustness of the model evaluation, the entire cross-validation process is repeated multiple times. In this case, it's repeated 3 times. Each repetition involves a new random splitting of the data into the same 10 folds, and the model is trained and evaluated on each fold in each repetition.

By using repeated cross-validation, you get a more reliable estimate of the model's performance because it averages the results over multiple runs, reducing the impact of randomness in the initial data splitting. This can provide a better assessment of how well the model is likely to perform on unseen data.

```r
knnPredict3 <- predict(knnFit3, newdata = test.data3)

# Creates the confusion matrix
confusionMatrix(data = knnPredict3, reference = test.data3$Diabetes_012)
```

```
## Confusion Matrix and Statistics
```

```
##
##           Reference
## Prediction   0   1
##          0 315 101
##          1 131 353
##
##                Accuracy : 0.7422
##                  95% CI : (0.7123, 0.7705)
##     No Information Rate : 0.5044
##     P-Value [Acc > NIR] : < 2e-16
##
##                   Kappa : 0.4841
##
##  Mcnemar's Test P-Value : 0.05692
##
##             Sensitivity : 0.7063
##             Specificity : 0.7775
##          Pos Pred Value : 0.7572
##          Neg Pred Value : 0.7293
##              Prevalence : 0.4956
##          Detection Rate : 0.3500
##    Detection Prevalence : 0.4622
##       Balanced Accuracy : 0.7419
##
##        'Positive' Class : 0
##
```

Model 3:

- Accuracy: 74.56%
- Kappa: 0.4907

Conclusions:

- Model 4, which was built using the predictors selected in **`train.data3`**, performs the best among all the models. It has the highest accuracy of 74.56% and the highest Kappa of 0.4907. This indicates that Model 4 is better at correctly classifying instances into either class 0 (no diabetes) or class 1 (diabetes).
- Model 1, Model 2, have very similar performance in terms of accuracy and Kappa, with differences that are not substantial. All three models have accuracy around 73.5% and Kappa around 0.47. This suggests that the predictors selected in **`train.data2`** (Model 1 and Model 2) are not significantly different in terms of predictive power.
- The Kappa statistic is a measure of agreement between the model's predictions and the actual classes. A higher Kappa value indicates better agreement beyond chance, which suggests a more reliable model.

Overall, Model 3 (with predictors selected in **`train.data3`**) is the preferred model due to its higher accuracy and Kappa. It's essential to select the model that provides the best balance between accuracy and reliability when making predictions.

KNN Models and Experiments to Find HeartDiseaseorAttack

Model 1:

```
###KNN Models and Experiments to Find HeartDiseaseorAttack ###############################################

## selection of 1500 samples of each factor of the dataset#
set.seed(1)
data_estratificada <- data %>%
  group_by(HeartDiseaseorAttack) %>%
  sample_n(1500, replace = TRUE) %>%
  ungroup()

predictors <- c("HighBP", "HighChol", "CholCheck", "BMI", "Smoker", "Stroke", "Diabetes_012", "PhysActi

# Original data
train.data <- data_estratificada[sample.index, c(predictors, "HeartDiseaseorAttack"), drop = FALSE]
test.data <- data_estratificada[-sample.index, c(predictors, "HeartDiseaseorAttack"), drop = FALSE]

train.data$HeartDiseaseorAttack <- factor(train.data$HeartDiseaseorAttack)
test.data$HeartDiseaseorAttack <- factor(test.data$HeartDiseaseorAttack)

# Train the k-NN model
ctrl <- trainControl(method = "cv", p = 0.7)
knnFit <- train(HeartDiseaseorAttack ~ .
                , data = train.data
                , method = "knn", trControl = ctrl
                , preProcess = c("range") # c("center", "scale") for z-score
                , tuneLength = 50)

# Make predictions
knnPredict <- predict(knnFit, newdata = test.data)

# Creates the confusion matrix
confusionMatrix(data = knnPredict, reference = test.data$HeartDiseaseorAttack)
```

Model 2:

```
### second model

predictors_to_remove <- c("AnyHealthcare", "NoDocbcCost", "DiffWalk", "Education", "Income")
train.data2 <- train.data[, !(names(train.data) %in% predictors_to_remove)]
test.data2 <- test.data[, !(names(test.data) %in% predictors_to_remove)]

# Train the k-NN model
ctrl <- trainControl(method = "cv", number = 5)
knnFit2 <- train(HeartDiseaseorAttack ~ .
                 , data = train.data2
                 , method = "knn", trControl = ctrl
                 , preProcess = c("range") # c("center", "scale") for z-score
                 , tuneLength = 50)


# Make predictions
```

```r
knnPredict2 <- predict(knnFit2, newdata = test.data2)

# Creates the confusion matrix
confusionMatrix(data = knnPredict2, reference = test.data2$HeartDiseaseorAttack)
```

Model 3:

```r
### Third Model

predictors_to_remove2 <- c("CholCheck", "MentHlth","HvyAlcoholConsump", "Fruits", "Veggies")
train.data3 <- train.data2[, !(names(train.data2) %in% predictors_to_remove2)]
test.data3 <- test.data2[, !(names(test.data2) %in% predictors_to_remove2)]

# Train the k-NN model
ctrl2 <- trainControl(method = "repeatedcv", number = 10, repeats = 3)
knnFit3 <- train(HeartDiseaseorAttack ~ .
                 , data = train.data3
                 , method = "knn", trControl = ctrl2
                 , preProcess = c("range") # c("center", "scale") for z-score
                 , tuneLength = 50)


# Make predictions
knnPredict3 <- predict(knnFit3, newdata = test.data3)

# Creates the confusion matrix
confusionMatrix(data = knnPredict3, reference = test.data3$HeartDiseaseorAttack)
```

Model 1:

- Accuracy: 74.56%
- Kappa: 0.4907

Model 2:

- Accuracy: 75.22%
- Kappa: 0.504

Model 3:

- Accuracy: 73.22%
- Kappa: 0.4639

Now, the analysis:

Accuracy: Model 2 has the highest accuracy (75.22%), followed by Model 1 (74.56%) and Model 3 (73.22%). Accuracy measures the proportion of correct predictions in the test set.

Kappa: Model 2 also has the highest Kappa value (0.504), followed by Model 1 (0.4907) and Model 3 (0.4639). Kappa evaluates the agreement between model predictions and actual classes while considering the possibility of random chance.

In summary, based on both accuracy and Kappa, Model 2 appears to be the best among the three models. Model 1 also performs well, and Model 3 is slightly less accurate. However, Model 2 is the top-performing model with the highest accuracy and Kappa, making it the recommended choice for this classification problem.

KNN Models and Experiments to Find Sex

Model 1:

```r
###KNN Models and Experiments to Find Sex #################################################################


## selection of 1500 samples of each factor of the dataset#
set.seed(1)
data_estratificada <- data %>%
  group_by(Sex) %>%
  sample_n(1500, replace = TRUE) %>%
  ungroup()

predictors <- c("HighBP", "HighChol", "CholCheck", "BMI", "Smoker", "Stroke", "HeartDiseaseorAttack" ,"I

# Original data
train.data <- data_estratificada[sample.index, c(predictors, "Sex"), drop = FALSE]
test.data <- data_estratificada[-sample.index, c(predictors, "Sex"), drop = FALSE]

train.data$Sex <- factor(train.data$Sex)
test.data$Sex <- factor(test.data$Sex)

# Train the k-NN model
ctrl <- trainControl(method = "cv", p = 0.7)
knnFit <- train(Sex ~ .
                , data = train.data
                , method = "knn", trControl = ctrl
                , preProcess = c("range") # c("center", "scale") for z-score
                , tuneLength = 50)


# Make predictions
knnPredict <- predict(knnFit, newdata = test.data)

# Creates the confusion matrix
confusionMatrix(data = knnPredict, reference = test.data$Sex)
```

Model 2:

```r
# second model

predictors_to_remove <- c("AnyHealthcare", "NoDocbcCost", "DiffWalk", "Age", "PhysActivity")
train.data2 <- train.data[, !(names(train.data) %in% predictors_to_remove)]
test.data2 <- test.data[, !(names(test.data) %in% predictors_to_remove)]

# Train the k-NN model
ctrl <- trainControl(method = "cv", number = 5)
```

```r
knnFit2 <- train(Sex ~ .
                 , data = train.data2
                 , method = "knn", trControl = ctrl
                 , preProcess = c("range") # c("center", "scale") for z-score
                 , tuneLength = 50)

#Make predictions
knnPredict2 <- predict(knnFit2, newdata = test.data2)

# Creates the confusion matrix
  confusionMatrix(data = knnPredict2, reference = test.data2$Sex)
```

Model 3:

```r
### Third Model

predictors_to_remove2 <- c("ChoclCheck", "MentHlth","HvyAlcoholConsump", "Fruits", "Veggies")
train.data3 <- train.data2[, !(names(train.data2) %in% predictors_to_remove2)]
test.data3 <- test.data2[, !(names(test.data2) %in% predictors_to_remove2)]

# Train the k-NN model
ctrl2 <- trainControl(method = "repeatedcv", number = 10, repeats = 3)
knnFit3 <- train(Sex ~ .
                 , data = train.data3
                 , method = "knn", trControl = ctrl2
                 , preProcess = c("range") # c("center", "scale") for z-score
                 , tuneLength = 50)


#Make predictions
knnPredict3 <- predict(knnFit3, newdata = test.data3)

# Creates the confusion matrix
confusionMatrix(data = knnPredict3, reference = test.data3$Sex)
```

The three models for predicting the sex of a person do not appear to be effective in this dataset. Here are the results for the three models:

Model 1:

- Accuracy: 57.56%
- Kappa: 0.1516

Model 2:

- Accuracy: 56.11%
- Kappa: 0.1227

Model 3:

- Accuracy: 54.89%

- Kappa: 0.0977

Analysis:

- The accuracy of all the models is quite low, ranging from 54.89% to 57.56%. This means that these models are not doing a good job of predicting a person's sex based on the provided features.
- The Kappa value is also very low in all models, indicating minimal agreement between the model's predictions and the actual sex of the individuals. Kappa takes into account agreement beyond what would be expected by chance, and these low values suggest that the models are not effective.
- In summary, the models are not suitable for predicting a person's sex based on the provided features in this dataset. This may be due to the complexity and lack of a clear relationship between the features and sex in this particular dataset.

Third part

Linear regression model BM

Model 1:

```r
### Linear regression model BMI #######################################################################
folder<-dirname(rstudioapi::getSourceEditorContext()$path)

parentFolder <-dirname(folder)

data <-
  read.csv(paste0(parentFolder,"/Dataset/diabetes_012.csv"))

data$Diabetes_012 <- ifelse(data$Diabetes_012 == 0, 0, 1)

set.seed(1)
data_estratificada2 <- data[sample(nrow(data), 3000), ]

predictors <- colnames(data_estratificada2)[-5]
sample.index <- sample(1:nrow(data_estratificada2),
                       nrow(data_estratificada2) * 0.7,
                       replace = FALSE)


train.data <- data_estratificada2[sample.index, c(predictors, "BMI"), drop = FALSE]
test.data <- data_estratificada2[-sample.index, c(predictors, "BMI"), drop = FALSE]

ins_model <- lm(BMI ~ ., data = train.data)

summary(ins_model)
```

```
##
## Call:
## lm(formula = BMI ~ ., data = train.data)
##
## Residuals:
##     Min      1Q  Median      3Q     Max
## -15.218  -3.753  -0.727   2.651  59.718
##
## Coefficients:
##                      Estimate Std. Error t value Pr(>|t|)
## (Intercept)          29.700892   1.248923  23.781  < 2e-16 ***
## Diabetes_012          2.282214   0.396631   5.754 1.00e-08 ***
## HighBP                2.821500   0.297689   9.478  < 2e-16 ***
## HighChol              0.566150   0.283749   1.995 0.046146 *
## CholCheck             0.859487   0.677266   1.269 0.204564
## Smoker               -0.522257   0.272625  -1.916 0.055546 .
## Stroke               -0.813064   0.708187  -1.148 0.251063
## HeartDiseaseorAttack -1.095276   0.483551  -2.265 0.023611 *
## PhysActivity         -0.974136   0.338502  -2.878 0.004046 **
## Fruits               -0.722677   0.287499  -2.514 0.012023 *
## Veggies              -0.537476   0.351942  -1.527 0.126870
## HvyAlcoholConsump    -0.945193   0.597458  -1.582 0.113796
## AnyHealthcare         0.162150   0.612766   0.265 0.791329
## NoDocbcCost          -0.528085   0.505369  -1.045 0.296168
## GenHlth               0.621751   0.163830   3.795 0.000152 ***
## MentHlth              0.006135   0.019977   0.307 0.758794
## PhysHlth             -0.049331   0.019188  -2.571 0.010210 *
## DiffWalk              2.097624   0.436809   4.802 1.68e-06 ***
## Sex                  -0.023210   0.274379  -0.085 0.932593
## Age                  -0.414443   0.048185  -8.601  < 2e-16 ***
## Education             0.065025   0.152360   0.427 0.669579
## Income               -0.113682   0.076249  -1.491 0.136132
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 5.968 on 2078 degrees of freedom
## Multiple R-squared:  0.157,  Adjusted R-squared:  0.1485
## F-statistic: 18.43 on 21 and 2078 DF,  p-value: < 2.2e-16
```

```
# Train the model
train.control <- trainControl(method = "cv", number = 10 )
model <- train(BMI ~ ., data = train.data, method = "lm",
               trControl = train.control)

# Summarize the results
print(model)
```

```
## Linear Regression
##
## 2100 samples
##   21 predictor
##
## No pre-processing
## Resampling: Cross-Validated (10 fold)
## Summary of sample sizes: 1891, 1891, 1890, 1889, 1889, 1891, ...
## Resampling results:
##
##   RMSE      Rsquared   MAE
##   5.984649  0.1486856  4.319634
##
```

```
## Tuning parameter 'intercept' was held constant at a value of TRUE
```

1. **Data Sampling:** The code begins by setting a random seed, then randomly selects 3000 rows from the dataset and stores it in `data_estratificada2`.
2. **Feature Selection:** The code selects all columns except the 5th column as predictors and stores them in the `predictors` variable. The 5th column ("BMI") is selected as the target variable for prediction.
3. **Train-Test Split:** The data is split into a training set (`train.data`) and a testing set (`test.data`) using a 70-30% split ratio. The training data will be used to build and train the linear regression model, and the testing data will be used to evaluate its performance.
4. **Initial Linear Regression Model:** An initial linear regression model (`ins_model`) is built using the training data to predict BMI based on all available predictors. A summary of this model is displayed.
5. **Model Training:** A linear regression model is trained using 10-fold cross-validation (`trainControl`) on the training data. This means the data is split into 10 subsets, and the model is trained and evaluated 10 times, each time using a different subset for validation.
6. **Results Summary:** The code prints a summary of the trained linear regression model (`model`).

Additionally, it's worth noting that no data normalization was performed in this code. Normalization was omitted because it didn't lead to improved results. Stratification of the sample was also avoided because it caused some variables, such as `Diabetes_012`, to behave abnormally in relation to BMI prediction. Therefore, it was decided not to stratify the sample as it did not provide meaningful improvements in this case.

Model 2:

```r
#### second

predictors_to_remove <- c("AnyHealthcare", "CholCheck", "MentHlth", "Education", "Sex")

train.data2 <- train.data[, !(names(train.data) %in% predictors_to_remove)]
test.data2 <- test.data[, !(names(test.data) %in% predictors_to_remove)]

ins_model <- lm(BMI ~ ., data = train.data2)

summary(ins_model)

# Train the model
train.control <- trainControl(method = "cv", number = 5)
model <- train(BMI ~ ., data = train.data2, method = "lm",
               trControl = train.control)

# Summarize the results
print(model)
```

Model 3:

```r
#### Third
predictors_to_remove <- c("Income", "Stroke", "NoDocbcCost", "Veggies", "HvyAlcoholConsump")

train.data3 <- train.data2[, !(names(train.data2) %in% predictors_to_remove)]
test.data3 <- test.data2[, !(names(test.data2) %in% predictors_to_remove)]
```

```r
ins_model <- lm(BMI ~ ., data = train.data3)

summary(ins_model)

# Train the model
train.control <- trainControl(method = "repeatedcv", number = 10, repeats = 3)
model <- train(BMI ~ ., data = train.data3, method = "lm",
               trControl = train.control)
# Summarize the results
print(model)
```

Model 1:

- RMSE (Root Mean Squared Error): 5.9846
- R-squared: 0.1487
- MAE (Mean Absolute Error): 4.3196

Model 2:

- RMSE (Root Mean Squared Error): 5.9871
- R-squared: 0.1456
- MAE (Mean Absolute Error): 4.3081

Model 3:

- RMSE (Root Mean Squared Error): 5.9483
- R-squared: 0.1514
- MAE (Mean Absolute Error): 4.3086

Comparing these models:

- The lowest RMSE is achieved by **Model 3** (RMSE = 5.9483), indicating that it has the smallest average prediction error.
- The highest R-squared value is also observed in **Model 3** (R-squared = 0.1514), which means it explains the most variance in the data.
- The lowest MAE is also obtained by **Model 3** (MAE = 4.3086), indicating that it has the smallest absolute prediction errors.

Based on these metrics, **Model 3** appears to be the best among the three models. It has the lowest RMSE, the highest R-squared value, and the lowest MAE, which collectively suggest that it provides the best predictive performance for predicting BMI.

Linear regression model MentHlth

Model 1

```r
### Linear regression model MentHlth ##################################################################

set.seed(1)
data_estratificada2 <- data[sample(nrow(data), 3000), ]

predictors <- colnames(data_estratificada2)[-16]
sample.index <- sample(1:nrow(data_estratificada2),
                       nrow(data_estratificada2) * 0.7,
                       replace = FALSE)

### ENTRENAMIENTO
train.data <- data_estratificada2[sample.index, c(predictors, "MentHlth"), drop = FALSE]
test.data <- data_estratificada2[-sample.index, c(predictors, "MentHlth"), drop = FALSE]

ins_model <- lm(MentHlth ~ ., data = train.data)
summary(ins_model)

# Train the model
train.control <- trainControl(method = "cv", number = 10 )
model <- train(MentHlth ~ ., data = train.data, method = "lm",
               trControl = train.control)

# Summarize the results
print(model)
```

Model 2

```r
###Second

predictors_to_remove <- c("BMI", "HeartDiseaseorAttack", "Stroke", "PhysActivity", "CholCheck")

train.data2 <- train.data[, !(names(train.data) %in% predictors_to_remove)]
test.data2 <- test.data[, !(names(test.data) %in% predictors_to_remove)]

ins_model <- lm(MentHlth ~ ., data = train.data2)
summary(ins_model)

# Train the model
train.control <- trainControl(method = "cv", number = 5)
model <- train(MentHlth ~ ., data = train.data2, method = "lm",
               trControl = train.control)

# Summarize the results
print(model)
```

Model 3

```r
#### Third
predictors_to_remove <- c("Diabetes_012", "HighBP", "HighChol", "Veggies", "Education")

train.data3 <- train.data2[, !(names(train.data2) %in% predictors_to_remove)]
test.data3 <- test.data2[, !(names(test.data2) %in% predictors_to_remove)]
```

```r
ins_model <- lm(MentHlth ~ ., data = train.data3)
summary(ins_model)

# Train the model
train.control <- trainControl(method = "repeatedcv", number = 10, repeats = 3)
model <- train(MentHlth ~ ., data = train.data3, method = "lm",
               trControl = train.control)
# Summarize the results
print(model)
```

Model 1:

- RMSE (Root Mean Squared Error): 6.5881
- R-squared: 0.1979
- MAE (Mean Absolute Error): 4.0371

Model 2:

- RMSE (Root Mean Squared Error): 6.5852
- R-squared: 0.1968
- MAE (Mean Absolute Error): 4.0267

Model 3:

- RMSE (Root Mean Squared Error): 6.5558
- R-squared: 0.2076
- MAE (Mean Absolute Error): 4.0103

Comparing these models:

- The lowest RMSE is achieved by **Model 3** (RMSE = 6.5558), indicating that it has the smallest average prediction error.
- The highest R-squared value is observed in **Model 3** (R-squared = 0.2076), meaning it explains the most variance in the data.
- The lowest MAE is also obtained by **Model 3** (MAE = 4.0103), indicating that it has the smallest absolute prediction errors.

Based on these metrics, **Model 3** appears to be the best among the three models for predicting `MentHlth`. It has the lowest RMSE, the highest R-squared value, and the lowest MAE, suggesting it provides the best predictive performance for this specific target variable.

In conclusion, the analysis of the three models for predicting `MentHlth` reveals that limiting the predictors to those with lower error metrics (such as RMSE, R-squared, and MAE) did indeed help improve the model's predictive performance.

Model 3, which had fewer predictors, outperformed the other models in terms of RMSE, R-squared, and MAE. This suggests that a more focused set of predictors, specifically selected based on their influence on the target variable, can lead to better predictive results.

Therefore, in this particular case, it's advantageous to limit the predictors to those that contribute more meaningfully to the prediction of `MentHlth`, as it results in a more accurate and effective predictive model.

Linear regression model PhysHlth

Model 1

```
#### Linear regression model PhysHlth #################################################################
set.seed(1)
data_estratificada3 <- data[sample(nrow(data), 3000), ]

predictors <- colnames(data_estratificada2)[-17]
sample.index <- sample(1:nrow(data_estratificada3),
                       nrow(data_estratificada3) * 0.7,
                       replace = FALSE)

train.data <- data_estratificada2[sample.index, c(predictors, "PhysHlth"), drop = FALSE]
test.data <- data_estratificada2[-sample.index, c(predictors, "PhysHlth"), drop = FALSE]

ins_model <- lm(PhysHlth ~ ., data = train.data)
summary(ins_model)

# Train the model
train.control <- trainControl(method = "cv", number = 10 )
model <- train(PhysHlth ~ ., data = train.data, method = "lm",
               trControl = train.control)
# Summarize the results
print(model)
```

Model 2

```
###Second

predictors_to_remove <- c("Sex", "Diabetes_012", "Education", "CholCheck", "Smoker")

train.data2 <- train.data[, !(names(train.data) %in% predictors_to_remove)]
test.data2 <- test.data[, !(names(test.data) %in% predictors_to_remove)]

ins_model <- lm(PhysHlth ~ ., data = train.data2)
summary(ins_model)

# Train the model
train.control <- trainControl(method = "cv", number = 5)
model <- train(PhysHlth ~ ., data = train.data2, method = "lm",
               trControl = train.control)
# Summarize the results
print(model)
```

Model 3

```
#### Third

predictors_to_remove <- c("BMI", "HeartDiseaseorAttack", "PhysActivity", "Veggies", "Stroke")

train.data3 <- train.data2[, !(names(train.data2) %in% predictors_to_remove)]
test.data3 <- test.data2[, !(names(test.data2) %in% predictors_to_remove)]
```

```r
ins_model <- lm(PhysHlth ~ ., data = train.data3)
summary(ins_model)

# Train the model
train.control <- trainControl(method = "repeatedcv", number = 10, repeats = 3)
model <- train(PhysHlth ~ ., data = train.data3, method = "lm",
               trControl = train.control)
# Summarize the results
print(model)
```

Model 1:

- RMSE: 6.858562
- R-squared: 0.4208921
- MAE: 4.594539

Model 2:

- RMSE: 6.872794
- R-squared: 0.4192887
- MAE: 4.589785

Model 3:

- RMSE: 6.902501
- R-squared: 0.4129561
- MAE: 4.662617

Now, considering these results, we can conclude the following:

- **Model 1** has an RMSE of 6.858562 and an R-squared of 0.4208921, indicating a good model fit, but there is still room for improvement.
- **Model 2** has similar results to Model 1, with an RMSE of 6.872794 and an R-squared of 0.4192887. Both models use a more limited set of predictors.
- **Model 3** shows a slight decrease in performance compared to Models 1 and 2, with a slightly higher RMSE of 6.902501 and an R-squared of 0.4129561.

In general, all models have comparable performance, but **Model 1** appears to be the most robust in terms of RMSE and R-squared. However, all three models are quite similar in terms of performance.

The main conclusion is that, by limiting predictors in Models 2 and 3, there was no significant improvement in performance compared to Model 1, which uses all available predictors. Therefore, in this case, including a broader set of predictors seems preferable as it does not significantly compromise predictive capability and may help capture more subtle relationships between predictor variables and the target variable `PhysHlth`.

To understand all the utilities of the model, please refer to:

- https://www.kaggle.com/datasets/alexteboul/diabetes-health-indicators-dataset
- https://www.kaggle.com/code/alexteboul/diabetes-health-indicators-dataset-
- notebook/notebook.https://www.cdc.gov/brfss/annual_data/2015/pdf/codebook15_llcp.pdf
- https://www.rdocumentation.org/packages/caret/versions/6.0-92/topics/trainControl
- http://scielo.sld.cu/scielo.php?pid=s1727-897x2010000200010&script=sci_arttext