kNN, Linear regression, and multilinear regression

Jennyfer Iglandini Torres Piraquive 84031

2023-10-08

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

Introduction

This report presents an exploratory analysis and predictive models performed on a health-related dataset "diabetes_012_health_indicators_BRFSS2015.csv". The analysis includes classification techniques using the K-NN algorithm and linear regression models to predict body mass index (BMI), mental health (MentHlth) and physical health (PhysHlth). Multiple experiments were conducted to assess the accuracy of the models and determine the most influential characteristics.

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

Download the libraries, (tidyverse) which promotes the concept of "ordered data", (caret) is a set of functions designed to streamline the process of creating predictive models, (class)(gmodels) and (psych), all to make the code to be created work well.

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

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

so:

- **`folder <- dirname(rstudioapi::getSourceEditorContext()$path)`**: Specifies the location in which the current script directory is located.
- **`parentFolder <- dirname(folder)`**:Gets the parent directory of the current directory.
- **`data <- read.csv(paste0(parentFolder,"/Dataset/diabetes_012.csv"))`**: Reads the CSV file named "diabetes_012.csv" located in the "Dataset" directory, and loads the data into the **`data`** variable. All these processes are linked one after the other

```r
data$Diabetes_012 <- ifelse(data$Diabetes_012 == 0, 0, 1)
set.seed(27)
data_ <- data[sample(nrow(data), 3000), ]

table(data_$Sex)
```

```
##
##    0    1
## 1685 1315
```

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

```
##
##    0    1
## 1635 1365
```

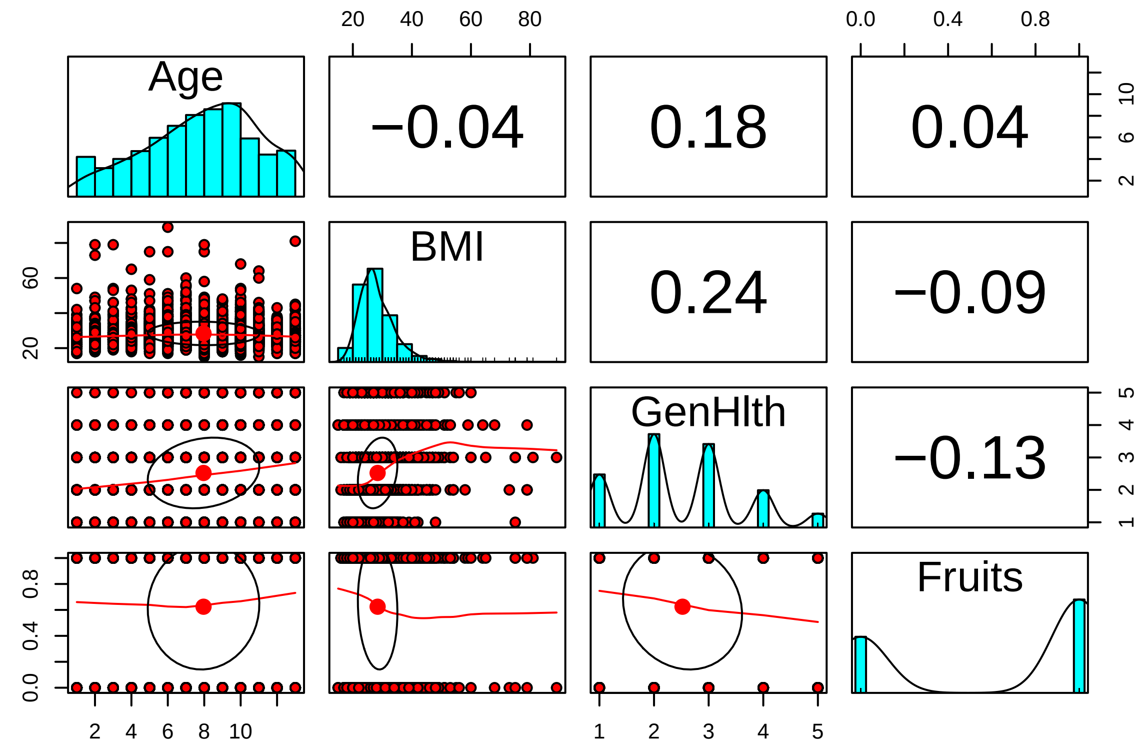
```r
table(data_$CholCheck)
```

```
##
##    0    1
##  113 2887
```

1. We 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 (27) to ensure reproducibility of the results and alignment with the report. A subset of the data is created to explore the data in a lighter and more efficient way. Next, we compute the frequency table for the variables “Sex”, “Smoker” and “CholCheck” in the “data” dataset.

2. From the previous data we can identify that the majority of the sample will be women and non-smokers, but it is expected that the group that has undergone cholesterol checks in the last 5 years is older. This process will give us an idea of the data set to interpret in future results.

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



El código **pairs.panels** se utiliza para crear una serie de diagramas de dispersión e histogramas que muestran las relaciones entre las variables seleccionadas y sus distribuciones. Aquí hay una explicación detallada del código:

- `data_[c("Edad", "IMC", "Educación", "GenHlth")]`: Selecciona las columnas "Edad", "IMC" (Índice de Masa Corporal), "Educación" y " GenHlth" (Salud general) del conjunto de datos `data_`, `pch = 21`: establece el tipo de punto en los diagramas de dispersión. En este caso se utiliza un punto en forma de círculo con un borde y `bg = c("red", "green3", "blue", "orange", "amarillo")[unclass(data_$Diabetes_012)]`: Define el color de fondo de los puntos en base a la Variable "Diabetes_012". Los puntos se colorearán dependiendo de si la variable "Diabetes_012" es 0 o 1. Regarding observations about the distributions:

The "Age" variable shows a normal distribution, meaning that the ages are fairly evenly distributed around the mean. This feature is important in certain statistical analyses, as some methods require data to be normally distributed to be valid.

On the other hand, BMI has a distribution skewed to the left, indicating that most people have lower Body Mass Index values. This suggests that there is a concentration of individuals with a lower BMI in the sample.

Second part KNN

KNN Models and Experiments to Find Diabetes

```
set.seed(27)
data_stratified <- data %>% group_by(Diabetes_012) %>%
  sample_n(1500, replace = TRUE) %>% ungroup()

sample.index <- sample(1:nrow(data_stratified)
                ,nrow(data_stratified)*0.75 ,replace = F)

predictors <- c("GenHlth", "MentHlth", "PhysHlth", "DiffWalk", "Sex", "Age", "Education", "Income","Hig

# data inicial
entrenamiento <- data_stratified[sample.index, c(predictors, "Diabetes_012"), drop = FALSE]
prueba<- data_stratified[-sample.index, c(predictors, "Diabetes_012"), drop = FALSE]

entrenamiento$Diabetes_012 <- factor(entrenamiento$Diabetes_012)
prueba$Diabetes_012 <- factor(prueba$Diabetes_012)
```

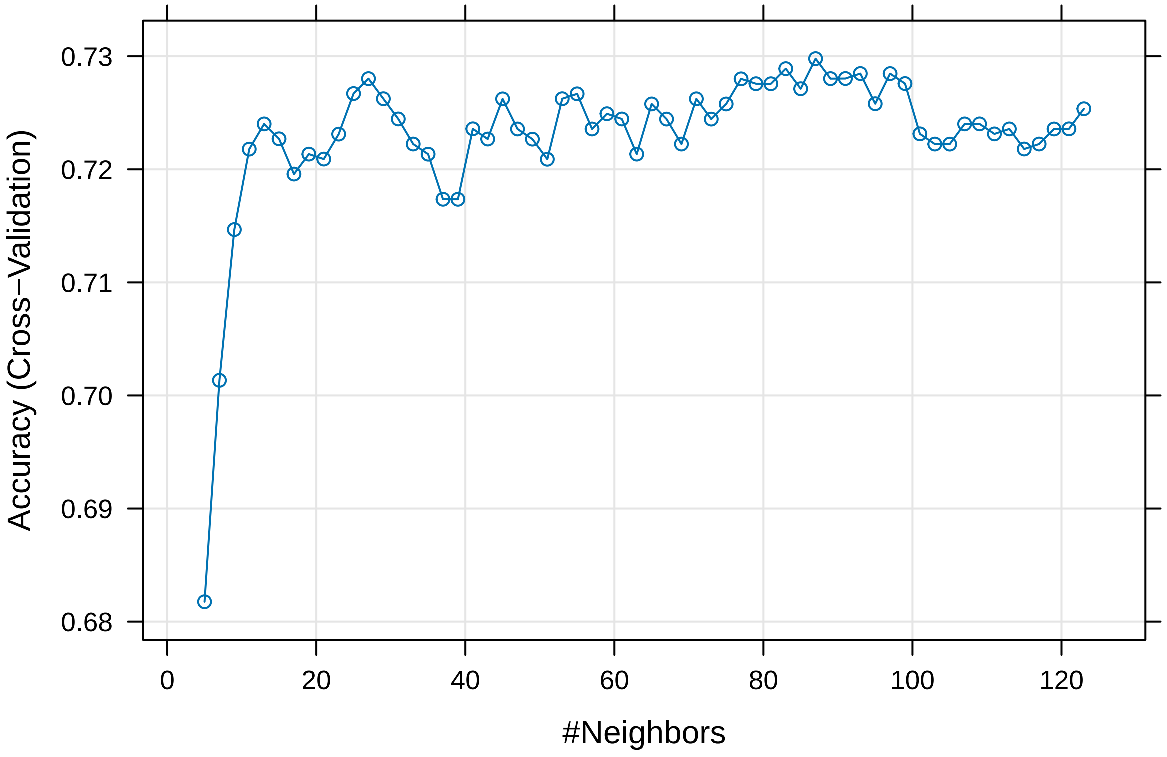
The first step is to prepare the data. At this stage all variables are considered except "Diabetes", which is the variable to be experimented with. To ensure consistent results, a specific random seed, set.seed(27), is defined.

The data is then divided into balanced subsets based on the variable "Diabetes_012," using a process called stratification. Of these subsets, 75% of the data is randomly chosen for training, identified as sample.index.

The predictor variables chosen for the analysis are specified and stored in the 'predictors' list. Then, the ¨entrenamiento¨ data set, called training, is created. Includes selected predictor variables and treats "Diabetes_012" as a categorical factor.

A similar approach is taken to construct the ¨prueba¨ set,test. This code prepares the data for machine learning by dividing it into training and test sets, while also selecting relevant predictor variables. Additionally, using a fixed random seed ensures that analysis results can be reproduced consistently.

```
control <- trainControl(method = "cv", p = 0.75)
knnFit <- train(Diabetes_012 ~ ., data = entrenamiento , method = "knn", trControl = control
        , preProcess = c("range") # c("center", "scale") for z-score
                , tuneLength = 60)
plot(knnFit)
```



First, the model is trained using the designated training data set (train.data), where "Diabetes_012" is the target variable and all available predictor variables are included (denoted by "." indicating all predictors).

TrainControl: In this step the control parameters for training the model are configured. Cross validation (method = "cv") is used with a 75% data partition assigned for training (p = 0.75).

Train: This function starts training the k-NN model with the specified parameters. It uses the training data set, uses the "knn" method and incorporates the previously defined control parameters. The tuneLength parameter is set to 60, indicating that 60 iterations will be performed to identify the optimal value of k.

plot(knnFit): This line of code generates a graph illustrating the performance of the trained k-NN model.

```
# Crear predicciones- confusion matrix
knnPredict <- predict(knnFit, newdata = prueba)
confusionMatrix(data = knnPredict, reference = prueba$Diabetes_012)
```

The predict function is used to predict the values of the variable "Diabetes_012" in new data using the pre-trained k-NN model (knnFit) and the test set (test.data). Predictions are saved in the knnPredict variable.

To evaluate the performance of the k-NN model, the confusionMatrix function is used to create a confusion matrix. This function evaluates how well the model predictions match the actual values and takes two arguments as input.

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction   0   1
##          0 252  91
##          1 122 285
##
##                Accuracy : 0.716
##                  95% CI : (0.6823, 0.748)
##     No Information Rate : 0.5013
##     P-Value [Acc > NIR] : < 2e-16
##
##                   Kappa : 0.4319
##
##  Mcnemar's Test P-Value : 0.03982
##
##             Sensitivity : 0.6738
##             Specificity : 0.7580
##          Pos Pred Value : 0.7347
##          Neg Pred Value : 0.7002
##              Prevalence : 0.4987
##          Detection Rate : 0.3360
##    Detection Prevalence : 0.4573
##       Balanced Accuracy : 0.7159
##
##        'Positive' Class : 0
##
```

The confusion matrix summarizes the performance of the model in classifying instances into two classes (0 and 1). In it, the rows represent the predicted classes (0 and 1), while the columns represent the actual classes (also 0 and 1). The four values in the matrix indicate True Negatives (TN, 252 instances correctly predicted as 0), False Positives (FP, 91 instances incorrectly predicted as 1), False Negatives (FN, 122 instances incorrectly predicted as 0), and True Positives (TP , 285 instances correctly predicted as 1). The accuracy of the model is 71.6%, which means that 71.6% of the predictions made by the model are correct, thus evaluating the overall accuracy of the model.

Kappa Index: The Kappa index (Kappa) is a measure of inter-rater agreement, in this case, the agreement between the model predictions and the actual classes. Adjust the precision based on the possibility of random agreement and take values between -1 and 1. In this case, Kappa is 0.4319

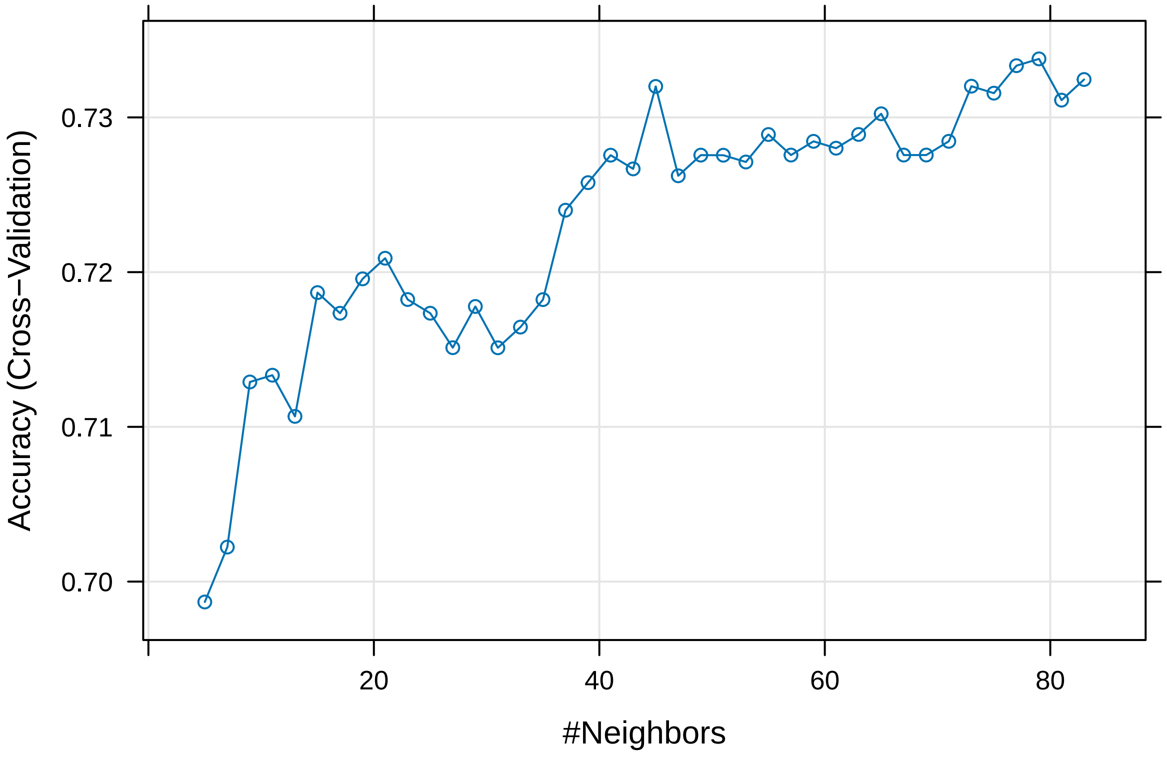
A Kappa of 1 indicates perfect agreement, a Kappa of 0 indicates agreement equivalent to chance agreement, a Kappa less than 0 indicates agreement worse than chance.

In this case, a Kappa of 0.4319 indicates moderate agreement between the model predictions and the actual classes. This value is not bad but it can be improved.

Second experiment

```r
predictors_to_remove <- c("AnyHealthcare", "NoDocbcCost", "DiffWalk", "Education", "Income")
entrenamiento2 <- entrenamiento[, !(names(entrenamiento) %in% predictors_to_remove)]
prueba2 <- prueba[, !(names(prueba) %in% predictors_to_remove)]

control <- trainControl(method = "cv", number = 5)
knnFit2 <- train(Diabetes_012 ~ ., data = entrenamiento2, method = "knn", trControl = control
          , preProcess = c("range") # c("center", "scale") for z-score
          , tuneLength = 40)
plot(knnFit2)
```



Analysis of second experiment

```
# Crear predicciones- confusion matrix
knnPredict2 <- predict(knnFit2, newdata = prueba2)
confusionMatrix(data = knnPredict2, reference = prueba2$Diabetes_012)
```

The confusion matrix summarizes the performance of the model in classifying instances into two classes (0 and 1). In it, the rows represent the predicted classes (0 and 1), while the columns represent the actual classes (also 0 and 1). The four values in the matrix indicate True Negatives (TN, 238 instances correctly predicted as 0), False Positives (FP, 76 instances incorrectly predicted as 1), False Negatives (FN, 136 instances incorrectly predicted as 0), and True Positives (TP , 300 instances correctly predicted as 1). The accuracy of the model is 71.73%, which means that 71.73% of the predictions made by the model are correct, thus evaluating the overall accuracy of the model. In this case, Kappa is 0.4344, a better value than in the previous experiment.

Sensitivity: Sensitivity, also known as the True Positive Rate or Recall, is 63.64%. It represents the proportion of actual positive cases (diabetes) correctly predicted by the model.

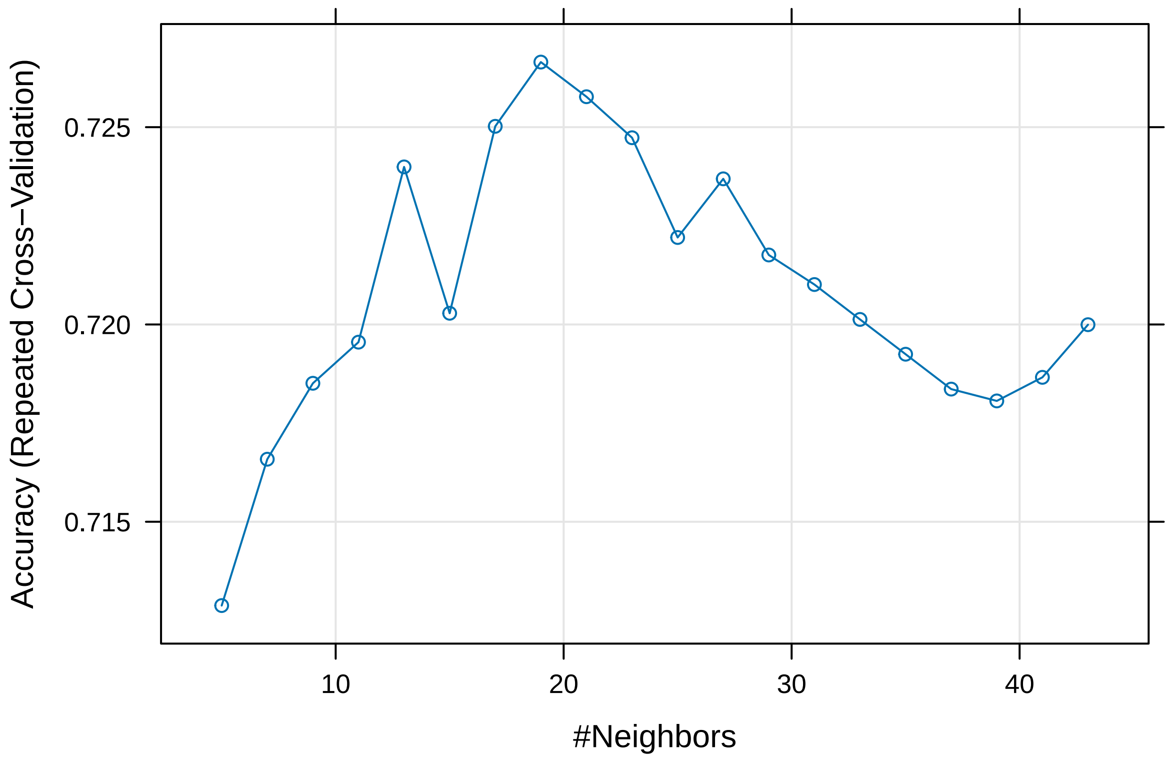
Specificity: Specificity is 79.79%, indicating the proportion of actual negative cases (no diabetes) correctly predicted by the model.

Tercer experimento

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 entrenamiento and prueba sets and averaging the results. Now excluding the “PhysHlth”, “Fruits”, “ChoclCheck”, “MentHlth”, “Veggies” variables, which I do not consider to contribute to the results.

```r
predictors_to_remove2 <- c("PhysHlth", "Fruits","CholCheck", "MentHlth","Veggies")
entrenamiento3 <- entrenamiento2[, !(names(entrenamiento2) %in% predictors_to_remove2)]
prueba3 <- prueba2[, !(names(prueba2) %in% predictors_to_remove2)]

control2 <- trainControl(method = "repeatedcv", number = 10, repeats = 3)
knnFit3 <- train(Diabetes_012 ~ ., data = entrenamiento3, method = "knn", trControl = control2
          , preProcess = c("range") # c("center", "scale") for z-score
          , tuneLength = 20)
plot(knnFit3)
```



Analysis of 3 experiment

```r
knnPredict3 <- predict(knnFit3, newdata = prueba3)
confusionMatrix(data = knnPredict3, reference = prueba3$Diabetes_012)
```

The confusion matrix summarizes the performance of the model in classifying instances into two classes (0 and 1). In it, the rows represent the predicted classes (0 and 1), while the columns represent the actual classes (also 0 and 1). The four values in the matrix indicate True Negatives (TN, 239 instances correctly predicted as 0), False Positives (FP, 81 instances incorrectly predicted as 1), False Negatives (FN, 135 instances incorrectly predicted as 0), and True Positives (TP , 295 instances correctly predicted as 1). The accuracy of the model is 71.2%, which means that 71.2% of the predictions made by the model are correct, thus evaluating the overall accuracy of the model.

Conclusion In model 2 a better Kappa is obtained (Kappa: 43.44%), also model 2 has the highest value in accuracy and specificity (71.73% and 79.79%, respectively), in terms of sensitivity model 1 is the best.

HeartDiseaseaseorAttack

For the model where the variable was **HeartDiseaseaseorAttack**, 3 experiments were done, which gave different results, in each experiment different variables were removed and the programming logic was the same as that of the **Diabetes_012** model. Obtaining as a result:

[[for more information about the code, lines 87 to 147, click on the following link: https://github.com/igtoj/JennyferTorres_A2/blob/main/R/Activity_A2.R]] **Modelo 1** All variables except the one to be analyzed were taken into account.

- Accuracy: 73.2%
- Kappa: 0.4628

Modelo 2 The variables ("AnyHealthcare", "NoDocbcCost", "DiffWalk", "Education", "Income") were disregarded.

- Accuracy: 74%
- Kappa: 0.4788

Modelo 3 The variables ("HvyAlcoholConsump", "Fruits","ChoclCheck", "MentHlth","Veggies") were disregarded.

- Accuracy: 74.67%
- Kappa: 0.4933

Model 2 also performs well and model 1 is slightly less accurate. However, model 3 obtains the best results with the highest accuracy and Kappa, making it the recommended choice for this classification problem.

Sex

For the model where the variable was **HeartDiseaseaseorAttack** and **Diabetes_012**, 3 experiments were done, which gave different results, in each experiment different variables were removed and the programming logic was the same as that of the **Sex** model. Obtaining as a result:

[[for more information about the code, lines 150 to 210, click on the following link: https://github.com/igtoj/JennyferTorres_A2/blob/main/R/Activity_A2.R]]

Modelo 1 All variables except the one to be analyzed were taken into account.

- Accuracy: 60.9%
- Kappa: 0.21.67

Modelo 2 The variables ("Age", "PhysActivity", "AnyHealthcare", "NoDocbcCost", "DiffWalk") were disregarded.

- Accuracy: 56.4%
- Kappa: 0.1302

Modelo 3 The variables("Fruits", "Veggies", "ChoclCheck", "MentHlth","HvyAlcoholConsump") were disregarded.

- Accuracy: 55.87%
- Kappa: 0.1191

Model 2 also performs well and model 3 is slightly less accurate. However, model 1 obtains the best results with the highest accuracy and Kappa, making it the recommended choice for this classification problem.

regresion lineal modelo BMI

This part of the code has aspects such as:

Data Sampling: 3000 rows are randomly selected from the dataset and stored in data_estratificada2.

Feature Selection: All columns except the 5th one are chosen as predictors and stored in the predictors variable. The 5th column ("BMI") is selected as the target variable.

Train-Test Split: The data is split into training (train.data) and testing (test.data) sets using a 70-30% ratio.

Initial Linear Regression Model: An initial linear regression model (ins_model) is created using the training data to predict BMI based on all predictors. A summary of this model is displayed.

Model Training: A linear regression model is trained using 10-fold cross-validation (trainControl) on the training data. The data is split into 10 subsets, and the model is trained and evaluated 10 times, each time using a different subset for validation.

```r
set.seed(27)
data_stratified2 <- data[sample(nrow(data), 3000), ]

predictors <- colnames(data_stratified2)[-5]
sample.index <- sample(1:nrow(data_stratified2),nrow(data_stratified2) * 0.75,
                replace = FALSE)


entrenamiento <- data_stratified2[sample.index, c(predictors, "BMI"), drop = FALSE]
prueba <- data_stratified2[-sample.index, c(predictors, "BMI"), drop = FALSE]

ins_model <- lm(BMI ~ ., data = entrenamiento)
summary(ins_model)
# entrenamiento del modelo
train.control <- trainControl(method = "cv", number = 10 )
model <- train(BMI ~ ., data = entrenamiento, method = "lm",
               trControl = train.control)
# Summarize the results
print(model)
```

Experiments 2 and 3 are done with the same programming base but the excluded variables are changed according to the model, in this case the variables of **BMI**, **MenHlth** and **PdysHlth** will be analyzed, obtaining as a result:

[[for more information about the code, lines 212 to 265, click on the following link: https://github.com/igtoj/JennyferTorres_A2/blob/main/R/Activity_A2.R]]

Model 1:

- RMSE (Root Mean Squared Error): 6.160693
- R-squared: 0.1307229
- MAE (Mean Absolute Error): 4.302126

Model 2:

- RMSE (Root Mean Squared Error): 6.15221
- R-squared: 0.1418412
- MAE (Mean Absolute Error): 4.300093

Model 3:

- RMSE (Root Mean Squared Error): 6.182873
- R-squared: 0.132676
- MAE (Mean Absolute Error): 4.295338

The lowest RMSE is achieved by *Model 2*, indicating that it has the smallest average prediction error.

The highest R-squared value is also observed in *Model 1*, which means it explains the most variance in the data.

The lowest MAE is also obtained by *Model 3*, indicating that it has the smallest absolute prediction errors.

According to these parameters, **Model 2** would be the best, since it apart from having the lowest RMSE, has the second lowest MAE, which together suggest that it provides the best predictive performance for predicting BMI.

Regresión lineal MentHlth

[[for more information about the code, lines 269 to 322, click on the following link: https://github.com/igtoj/JennyferTorres_A2/blob/main/R/Activity_A2.R]]

In this model the variable **MentHlth** will be analyzed, it was obtained as a result in each model (experiment):

Model 1:

- RMSE (Root Mean Squared Error): 6.618564
- R-squared: 0.1903101
- MAE (Mean Absolute Error): 4.039904

Model 2:

- RMSE (Root Mean Squared Error): 6.590597
- R-squared: 0.196061
- MAE (Mean Absolute Error): 4.019626

Model 3:

- RMSE (Root Mean Squared Error): 6.618564
- R-squared: 0.1950633
- MAE (Mean Absolute Error): 4.014207

The lowest RMSE is achieved by *Model 2*, indicating that it has the smallest average prediction error.

The highest R-squared value is also observed in *Model 1*, which means it explains the most variance in the data.

The lowest MAE is also obtained by *Model 3*, indicating that it has the smallest absolute prediction errors.

According to these parameters, **Model 2** would be the best, since it apart from having the lowest RMSE, has the second lowest MAE, which together suggest that it provides the best predictive performance for predicting BMI.

Linear regression model PhysHlth [[for more information about the code, lines 325 to 378, click on the following link: https://github.com/igtoj/JennyferTorres_A2/blob/main/R/Activity_A2.R]]

In this model the variable **PhysHlth** will be analyzed, it was obtained as a result in each model (experiment):

Model 1:

- RMSE (Root Mean Squared Error): 6.961776
- R-squared: 0.3904411
- MAE (Mean Absolute Error): 4.676039

Model 2:

- RMSE (Root Mean Squared Error):6.963018
- R-squared: 0.3958256
- MAE (Mean Absolute Error): 4.65453

Model 3:

- RMSE (Root Mean Squared Error): 6.965291
- R-squared: 0.39296
- MAE (Mean Absolute Error): 4.683361

The lowest RMSE is achieved by *Model 1*, indicating that it has the smallest average prediction error.

The highest R-squared value is also observed in *Model 1*, which means it explains the most variance in the data.

The lowest MAE is also obtained by *Model 2*, indicating that it has the smallest absolute prediction errors.

According to these parameters, **Model 1** would be the best, since it apart from having the lowest RMSE, has the second lowest MAE, which together suggest that it provides the best predictive performance for predicting BMI.

Thank you very much :3