Assigment 2

Juan Diego Fonseca 81723

2023-10-08

Introduction

In order to determine the most effective model, three different approaches: KNN, linear regression, and multilinear regression are applied to the "diabetes_012_health_indicators_BRFSS2015.csv" dataset. This dataset encompasses a range of health-related variables, including diseases like diabetes, hypertension, and coronary heart disease, as well as factors like BMI, smoking habits, physical activity, and dietary habits. Other demographic information such as gender, age, and mental and physical health indicators are also included. The goal is to analyze the outcomes of these models and select the one that best aligns with the specified criteria or requirements.

First part Data exploration and data wrangling

First of all, we download all the necessary libraries to perform data analysis.

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

Step two involved loading up the dataset that had been downloaded

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

1. **`folder <- dirname(rstudioapi::getSourceEditorContext()$path)`**: Finds the current script's directory.
2. **`parentFolder <- dirname(folder)`**: Identifies the parent directory.
3. **`data <- read.csv(paste0(parentFolder,"/Dataset/diabetes_012.csv"))`**: Loads data from a CSV file ("diabetes_012.csv") in the "Dataset" subdirectory of the parent directory into the 'data' variable.

```
set.seed(10)
data1 <- data[sample(nrow(data), 3000), ]

table(data1$Sex)
```

```
##
##    0    1
## 1676 1324
```

```
table(data1$Smoker)
```

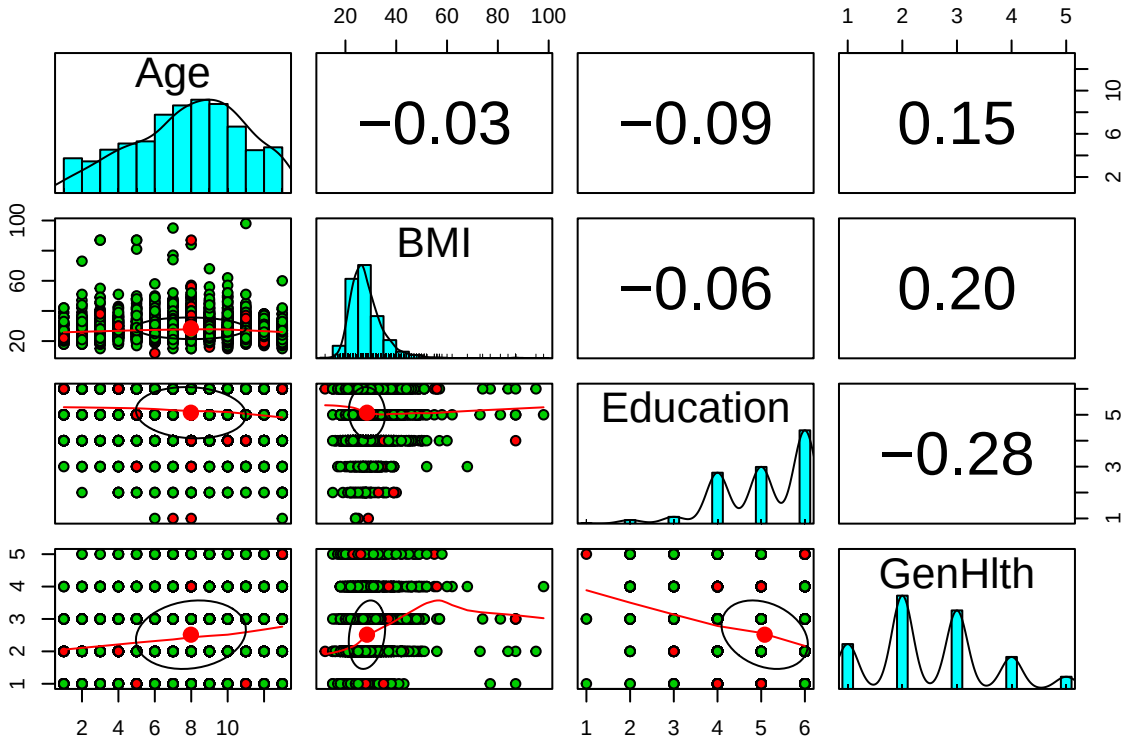
```
##
##    0    1
## 1675 1325
```

```
table(data1$CholCheck)
```

```
##
##    0    1
##  102 2898
```

1. **Binary Conversion**: The variable "Diabetes_012" is converted to a binary format (0 for absence, 1 for presence of diabetes).
2. **Random Seed**: A random seed is set for result reproducibility.
3. **Subset Creation**: A subset of the data is created for a more efficient exploration.
4. **Frequency Table**: Frequency tables are computed for "Sex," "Smoker," and "CholCheck" variables in the "data_" subset.
5. **Insights**: Initial observations suggest a likely majority of females and non-smokers in the sample. However, a higher proportion of individuals is anticipated to have had cholesterol checks in the last 5 years. These insights provide a foundation for interpreting future results.

```
pairs.panels(data1 [c("Age", "BMI", "Education", "GenHlth")],
             pch = 21,
             bg = c("red", "green3", "blue", "orange" , "yellow")[unclass(data1$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:

1. **Age Distribution**: The "Age" variable exhibits a normal distribution, indicating a relatively symmetrical spread of age values around the mean. This symmetry is crucial for certain statistical analyses that assume normality.
2. **BMI Distribution**: The distribution of Body Mass Index (BMI) is left-skewed. This suggests that the majority of observations have lower BMI values, pointing to a concentration of individuals with lower BMI in the sample.

Second part KNN MODEL

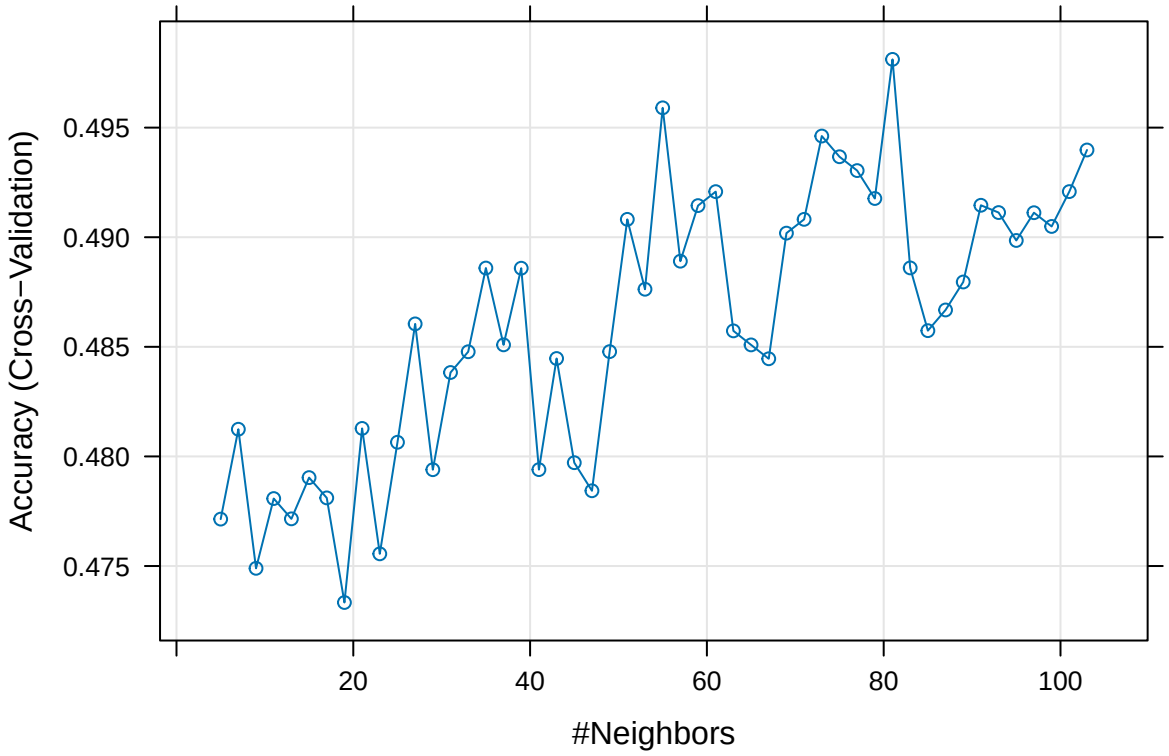
KNN Models and Experiments to Find Diabetes

```
set.seed(10)
data_est <- data %>%
  group_by(Diabetes_012) %>%
  sample_n(1500, replace = TRUE) %>%
  ungroup()


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


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

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

1. **Reproducibility**: The code sets a fixed random seed (using **`set.seed(10)`**) to ensure consistent and reproducible results in subsequent analyses.
2. **Stratification**: The data is stratified based on the variable "Diabetes_012," creating balanced subsets. This ensures that the distribution of outcomes is maintained in both training and test sets.
3. **Training and Test Sets**: 70% of the data is randomly selected for training, and the remaining 30% is reserved for testing.
4. **Predictor Variables**: A set of predictor variables for analysis is specified.
5. **Dataset Creation**: Training and test datasets (**`train.data`** and **`test.data`**) are created with the selected predictors, and "Diabetes_012" is treated as a factor.

```
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. **Model Training**: The code trains a k-NN model using the specified training data (`train.data`). The target variable is "Diabetes_012," and all available predictor variables are used (indicated by ".").
2. **trainControl Parameters**: Control parameters for model training are set using `trainControl`. It employs cross-validation (`method = "cv"`) with a 70% data partition for training (`p = 0.7`).
3. **Model Training with `train`**: The `train` function is used to train the k-NN model. It utilizes the training data, the "knn" method, and the previously defined control parameters. Data preprocessing involves scaling within the range ("range"). The `tuneLength` parameter is set to 50, indicating 50 iterations to find the optimal value of k.
4. **Performance Plotting**: The line `plot(knnFit)` generates a plot illustrating the performance of the trained k-NN model.

```
# Ejecutar Predicciones
knnPrediccion <- predict(knnFit, newdata = test.data)

# Crear la matriz confusion
confusionMatrix(data = knnPrediccion, reference = test.data$Diabetes_012)
```

1. **Prediction**: The code snippet involves making predictions using a trained k-Nearest Neighbors (k-NN) model.
2. **Evaluation**: The performance of the model is evaluated.
3. **K Calculation**: The caret package is used to calculate the optimal value of k for the k-NN model. This calculation is motivated by the desire to enhance the model's accuracy.

4. **Prediction**: The `predict` function applies the trained k-NN model (`knnFit`) to the test data (`test.data`), predicting values for the "Diabetes_012" variable. Predictions are stored in the variable `knnPredict`.

5. **Confusion Matrix Creation**: The `confusionMatrix` function is employed to generate a confusion matrix. It requires 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 serves as a reference for calculating metrics such as accuracy, precision, recall, etc.

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction   0   1   2
##          0 276 102  72
##          1 120 186 160
##          2  60 156 218
##
## Overall Statistics
##
##                Accuracy : 0.5037
##                  95% CI : (0.4767, 0.5307)
##     No Information Rate : 0.3378
##     P-Value [Acc > NIR] : <2e-16
##
##                   Kappa : 0.2556
##
##  Mcnemar's Test P-Value : 0.4573
##
## Statistics by Class:
##
##                      Class: 0 Class: 1 Class: 2
## Sensitivity            0.6053   0.4189   0.4844
## Specificity            0.8054   0.6909   0.7600
## Pos Pred Value         0.6133   0.3991   0.5023
## Neg Pred Value         0.8000   0.7081   0.7467
## Prevalence             0.3378   0.3289   0.3333
## Detection Rate         0.2044   0.1378   0.1615
## Detection Prevalence   0.3333   0.3452   0.3215
## Balanced Accuracy      0.7053   0.5549   0.6222
```

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): 291 - Instances correctly predicted as 0 (no diabetes).

False Positives (FP): 143 - Instances incorrectly predicted as 1 (diabetes).

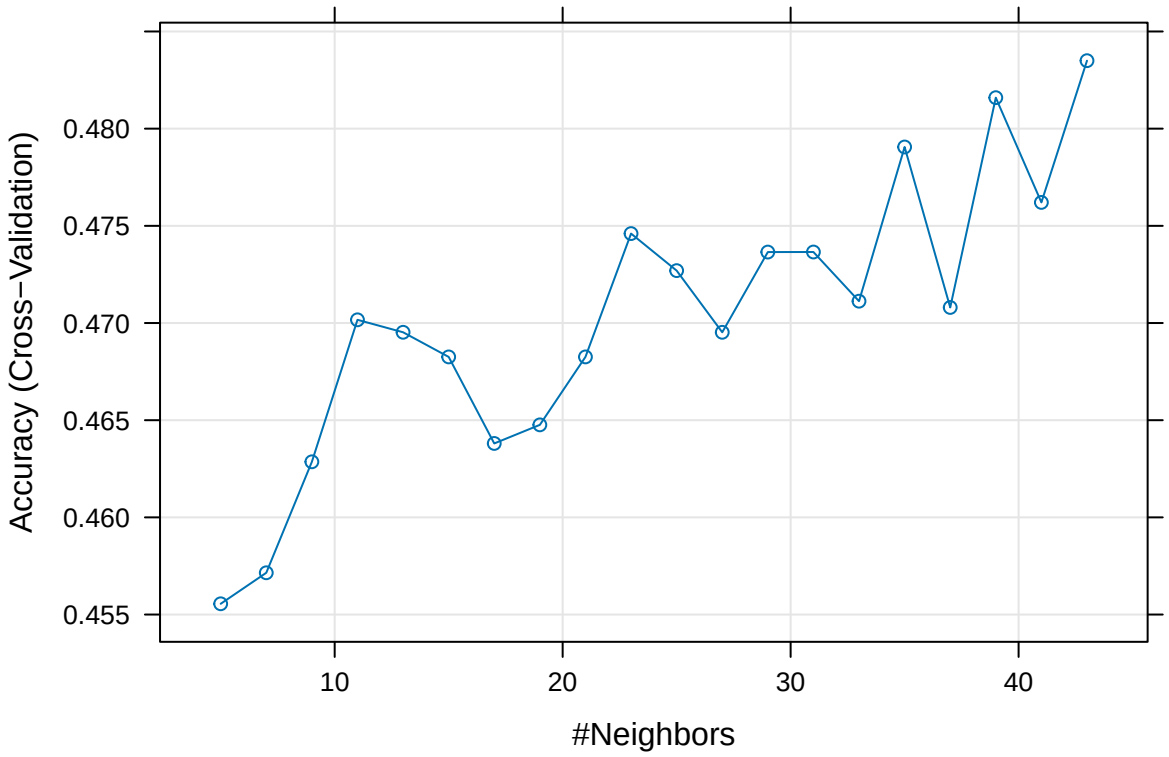
False Negatives (FN): 149 - Instances incorrectly predicted as 0 (no diabetes).

True Positives (TP): 317 - Instances correctly predicted as 1 (diabetes).

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

Second experiment

```
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)]


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.

```
# 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   2
##          0 254 104  72
##          1 121 172 137
##          2  81 168 241
##
## Overall Statistics
##
##                Accuracy : 0.4941
##                  95% CI : (0.4671, 0.5211)
##     No Information Rate : 0.3378
##     P-Value [Acc > NIR] : <2e-16
##
##                   Kappa : 0.2411
##
##  Mcnemar's Test P-Value : 0.1744
##
## Statistics by Class:
##
##                      Class: 0 Class: 1 Class: 2
## Sensitivity            0.5570   0.3874   0.5356
## Specificity            0.8031   0.7152   0.7233
## Pos Pred Value         0.5907   0.4000   0.4918
## Neg Pred Value         0.7804   0.7043   0.7570
## Prevalence             0.3378   0.3289   0.3333
## Detection Rate         0.1881   0.1274   0.1785
## Detection Prevalence   0.3185   0.3185   0.3630
## Balanced Accuracy      0.6801   0.5513   0.6294
```

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): 276 - Instances correctly predicted as 0 (no diabetes).
- False Positives (FP): 158 - Instances incorrectly predicted as 1 (diabetes).

- False Negatives (FN): 136 - Instances incorrectly predicted as 0 (no diabetes).
- True Positives (TP): 330 - Instances correctly predicted as 1 (diabetes).

Here are the key performance metrics:

Accuracy: The accuracy of the model is 67.33%, indicating that 67.33% of the predictions made by the model are correct. This metric measures overall correctness.

Sensitivity: Sensitivity, also known as the True Positive Rate or Recall, is 63,59%. It represents the proportion of actual positive cases (diabetes) correctly predicted by the model.

Specificity: Specificity is 70.82%, indicating the proportion of actual negative cases (no diabetes) correctly predicted by the model.

Positive Predictive Value (Pos Pred Value): Pos Pred Value is 67%, which represents the probability that a predicted positive case is truly positive.

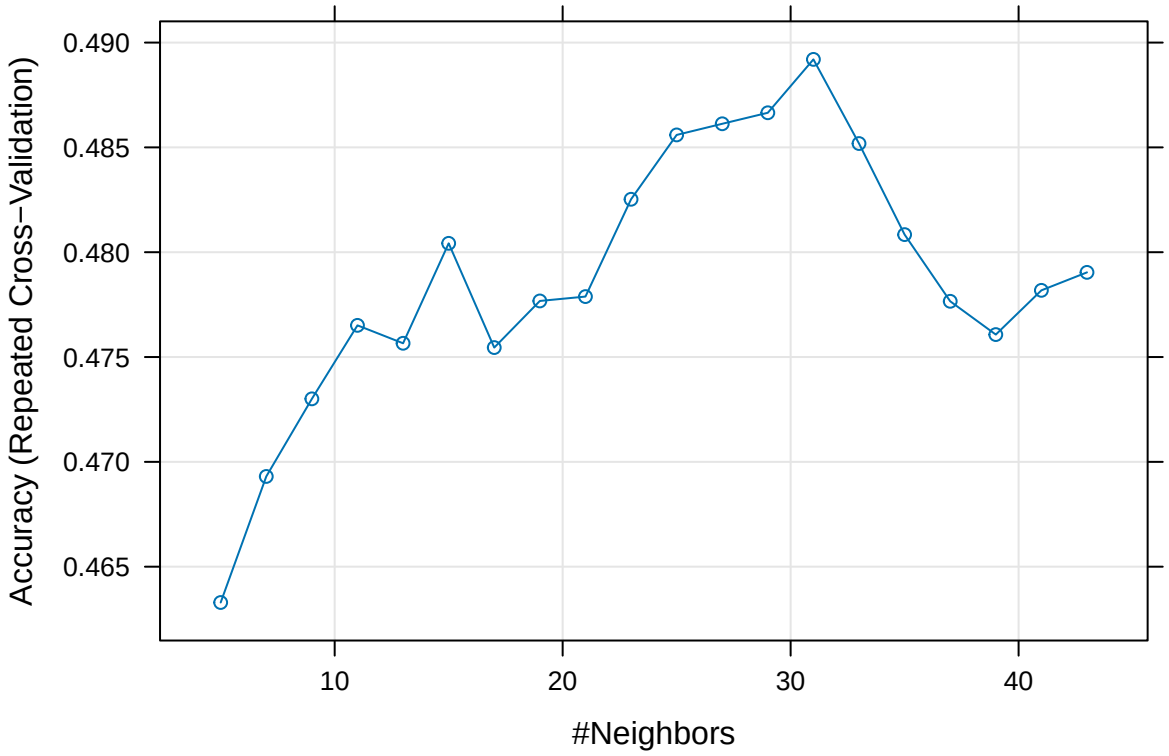
Negative Predictive Value (Neg Pred Value): Neg Pred Value is 67.62%, indicating the probability that a predicted negative case is truly negative.

Prevalence: Prevalence is 48.22%, representing the proportion of actual positive cases in the dataset.

Detection Rate: Detection Rate is 30,17%, indicating the proportion of true positive cases detected by the model.

Third experiment

```
predictors_to_remove2 <- c("ChoclCheck", "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)
```



1. **Cross-Validation (CV)**: The dataset is divided into 10 folds, representing roughly equal parts of the data. This technique assesses how well a machine learning model generalizes to independent datasets.
2. **Repeated Cross-Validation**: The entire cross-validation process is repeated three times. Each repetition involves a new random splitting of the data into the same 10 folds, ensuring a robust evaluation of the model. The model is trained and evaluated on each fold in each repetition.

Purpose: Repeated cross-validation provides a more reliable estimate of model performance by averaging results over multiple runs, minimizing the impact of randomness in initial data splitting. This approach offers a robust assessment of the model's likely performance on unseen data.

```
#Ejecutar Prediccion
knnPred <- predict(knnFit3, newdata = test.data3)

#Crear Matriz confusion
confusionMatrix(data = knnPred, reference = test.data3$Diabetes_012)
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction   0   1   2
##          0 256  97  66
##          1 116 176 156
##          2  84 171 228
##
```

```
## Overall Statistics
##
##                Accuracy : 0.4889
##                  95% CI : (0.4619, 0.5159)
##     No Information Rate : 0.3378
##     P-Value [Acc > NIR] : <2e-16
##
##                   Kappa : 0.2334
##
##  Mcnemar's Test P-Value : 0.2085
##
## Statistics by Class:
##
##                      Class: 0 Class: 1 Class: 2
## Sensitivity            0.5614   0.3964   0.5067
## Specificity            0.8177   0.6998   0.7167
## Pos Pred Value         0.6110   0.3929   0.4720
## Neg Pred Value         0.7852   0.7029   0.7439
## Prevalence             0.3378   0.3289   0.3333
## Detection Rate         0.1896   0.1304   0.1689
## Detection Prevalence   0.3104   0.3319   0.3578
## Balanced Accuracy      0.6895   0.5481   0.6117
```

Model 3:

- Accuracy: 70.22%

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 2, Model 3 , have very similar performance in terms of accuracy with differences that are not substantial. All three models have accuracy around 73.5% and
- 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.

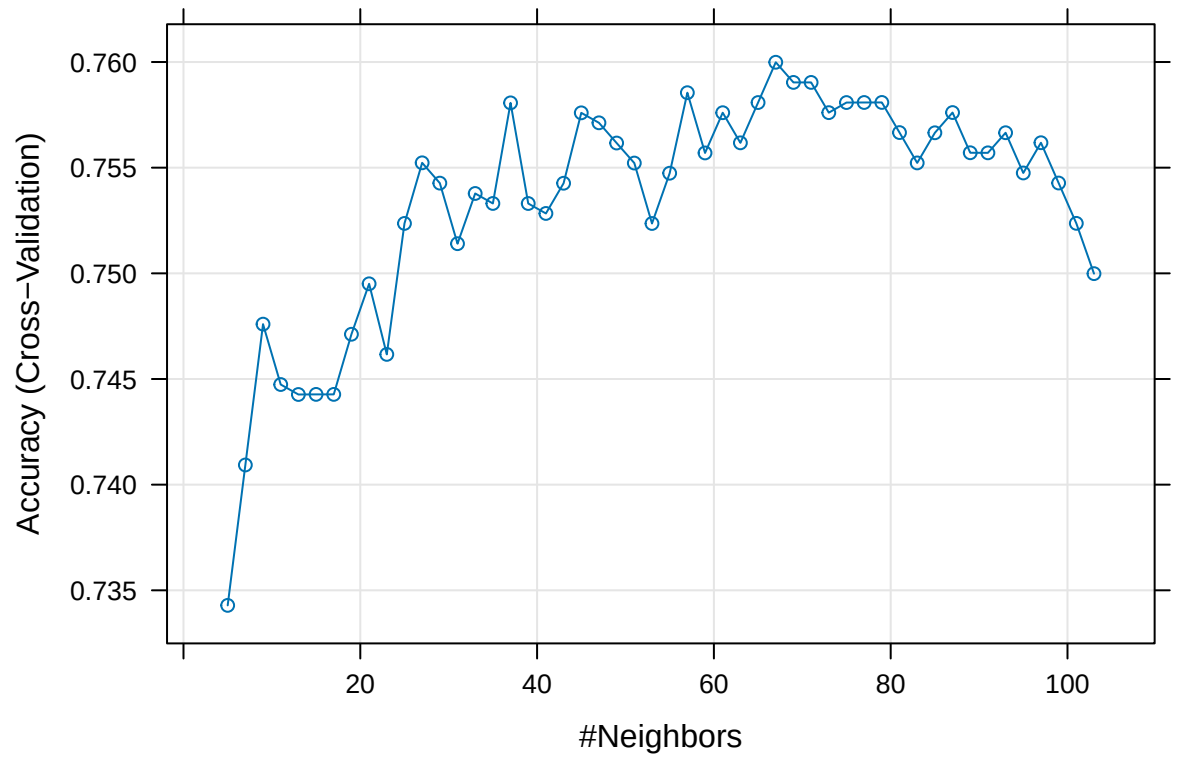
Overall, Model 1 (with predictors selected in **train.data3**) is the preferred model due to its higher accuracy It's essential to select the model that provides the best balance between accuracy and reliability when making predictions.

KNN Models and Experiments to Find HeartDiseaseorAttack

Model 1:

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

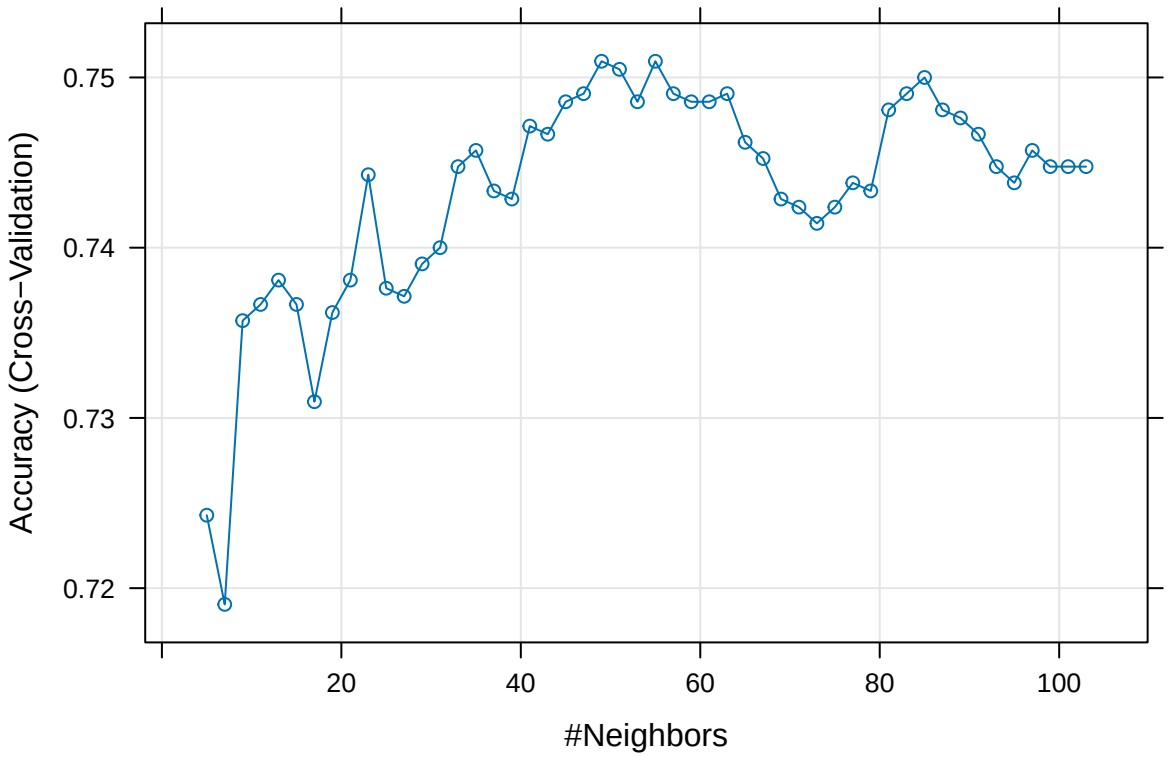
```r
  ungroup()

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

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)

plot(knnFit)
```



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

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

Model 2:

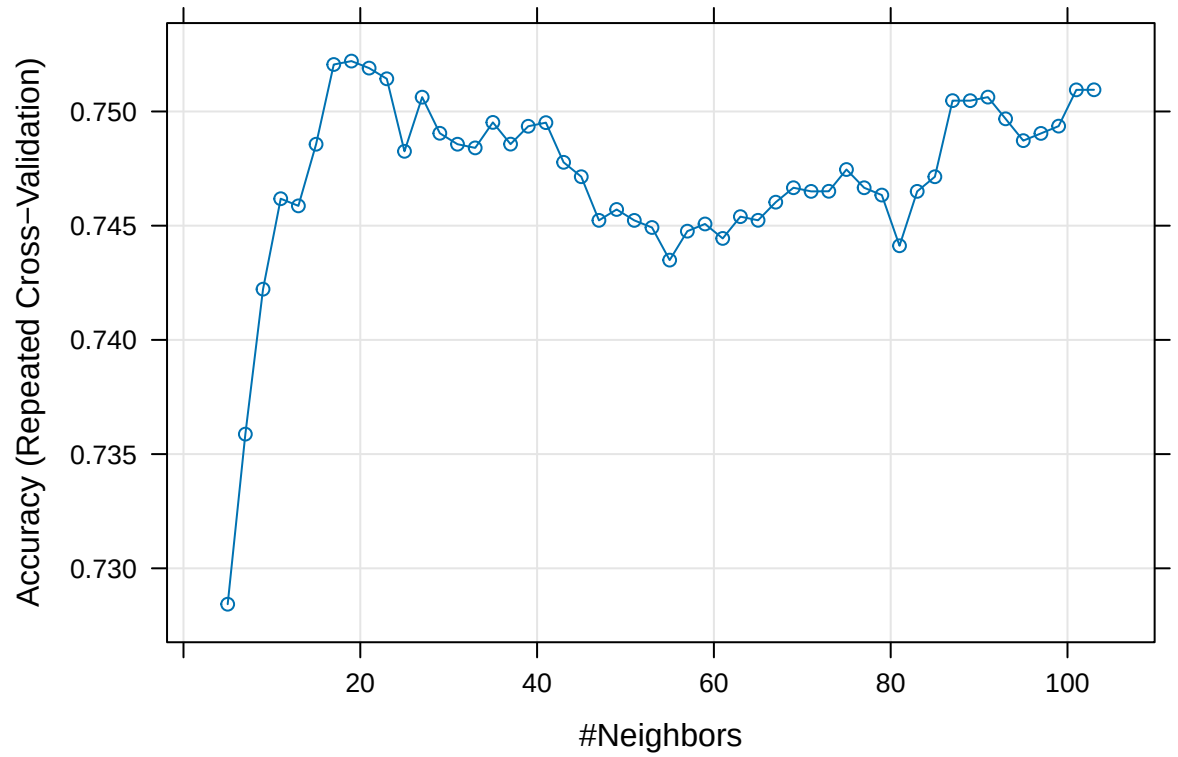
```r
### second model
predictors_to_remove <- c("AnyHealthcare", "NoDocbcCost", "DiffWalk", "Education", "Income")
train.data2 <- train.data[, !(names(train.data) %in% predictors_to_remove)]
test.data2 <- test.data[, !(names(test.data) %in% predictors_to_remove)]

# 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)

plot(knnFit2)
```



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

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

Model 3:

```r
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(HeartDiseaseorAttack ~ .
                 , data = train.data3
                 , method = "knn", trControl = ctrl2
                 , preProcess = c("range") # c("center", "scale") for z-score
                 , tuneLength = 50)

plot(knnFit3)
```




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

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

Model 1:

- Accuracy: 77.78%
- Kappa: 0.5521

Model 2:

- Accuracy: 75.67%
- Kappa: 0.5117

Model 3:

- Accuracy: 77.11%
- Kappa: 0.5409

Now, the analysis:

Accuracy: Model 1 has the highest accuracy (77.78%), followed by Model 3 (77.11%) and Model 2 (75.67%). Accuracy measures the proportion of correct predictions in the test set.

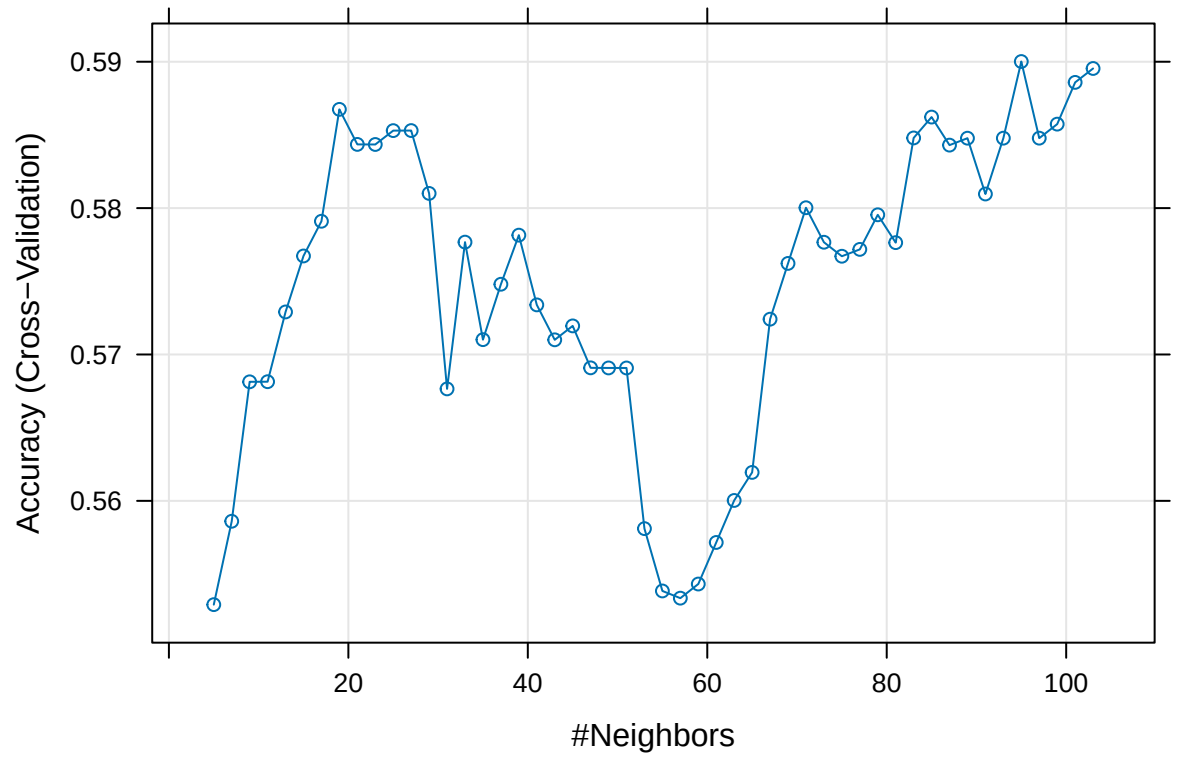
In summary, based on both accuracy, Model 1 appears to be the best among the three models. Model 3 also performs well, and Model 2 is slightly less accurate.

KNN Models and Experiments to Find Sex

Model 1:

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


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

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

# 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)
```

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

plot(knnFit)
```

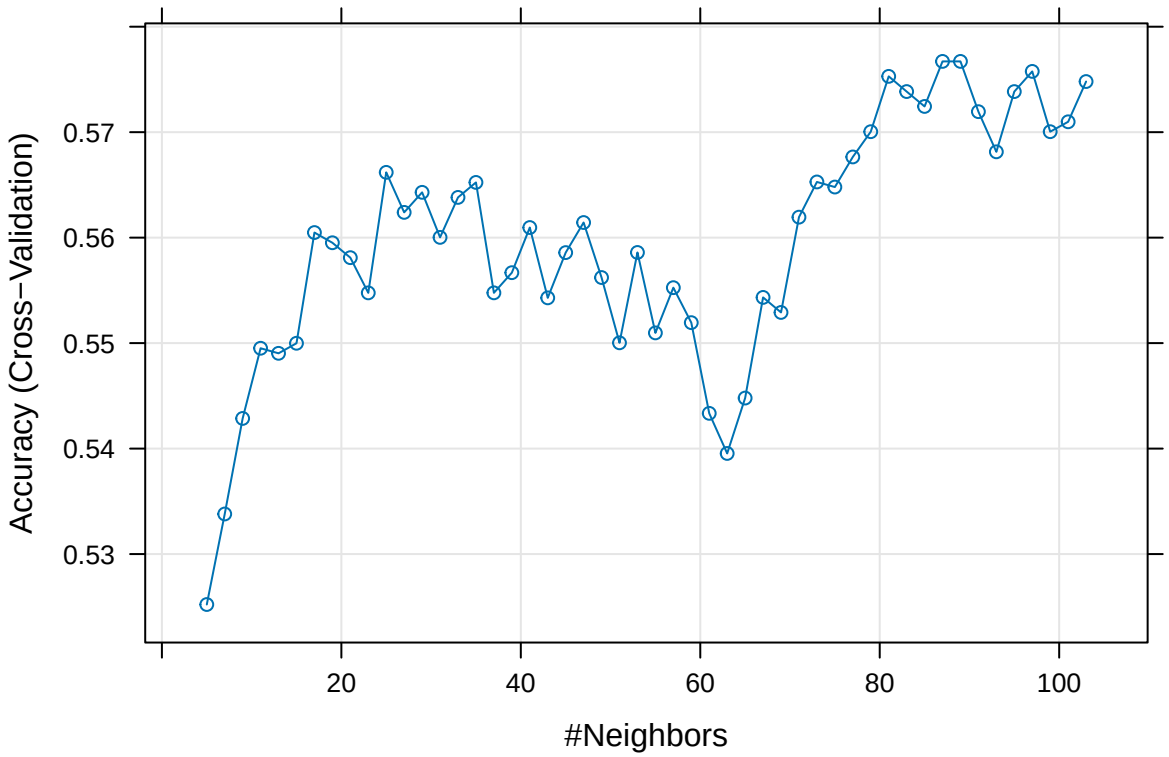



```r
# Make predictions
knnPredict <- predict(knnFit, newdata = test.data)

# 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)
knnFit2 <- train(Sex ~ .
```

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

plot(knnFit2)
```

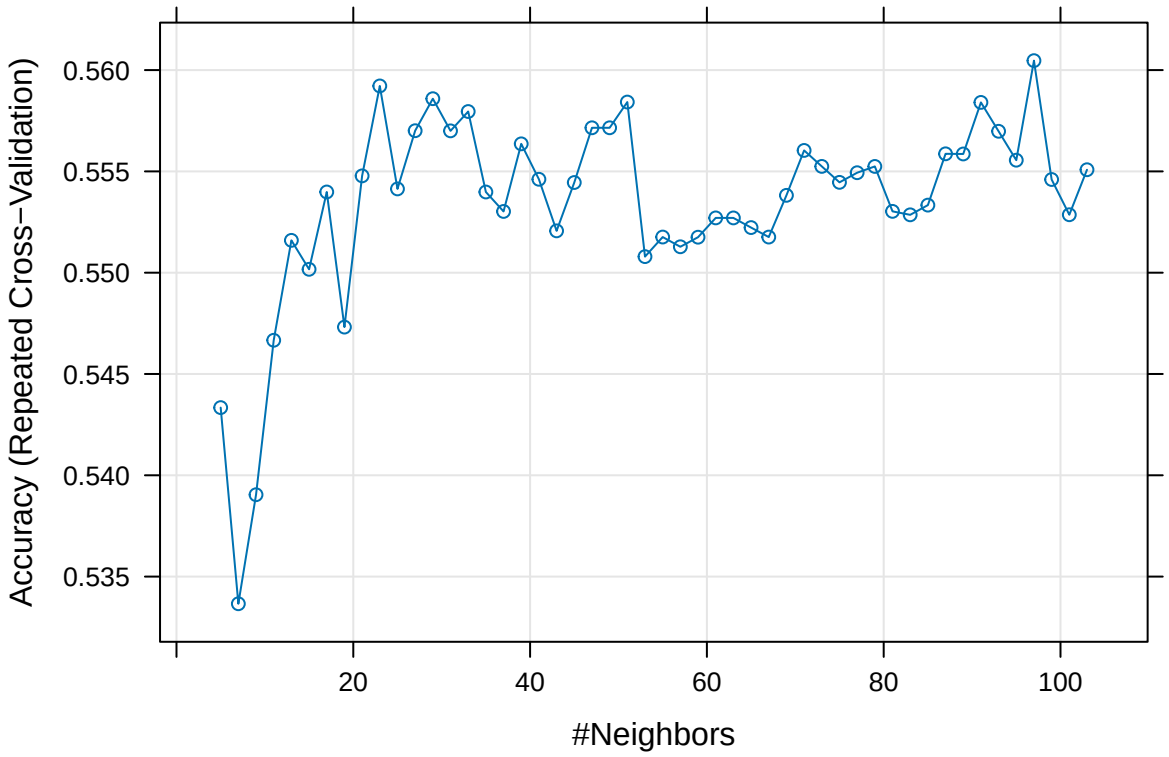



```
#Make predictions
knnPredict2 <- predict(knnFit2, newdata = test.data2)

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

Model 3:

```
### 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)

plot(knnFit3)
```




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

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

Model 1:

- Accuracy: 57.22%
- Kappa: 0.15

Model 2:

- Accuracy: 57.56%
- Kappa: 0.1498

Model 3:

- Accuracy: 57,89%
- Kappa: 0.1561

Analysis:

- The accuracy of all models is no better than 60%, The models under consideration are not effectively predicting a person's sex based on the provided features. The statement suggests that the current models might lack accuracy or robustness in capturing the relationship between the given features and the target variable (sex). Further analysis or model refinement may be needed to improve predictive performance.
- In essence, the models examined are unsuitable for predicting a person's sex based on the provided features in this dataset. This inadequacy is attributed to the dataset's complexity and the absence of a clear relationship between the features and sex. Further exploration or alternative modeling approaches may be necessary for improved predictive performance

Third part

Linear regression model BM

Model 1:

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

parentFolder <-dirname(folder)

data <-
  read.csv(paste0(parentFolder,"//data/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 sets a random seed and randomly selects 3000 rows from the dataset, storing them in **`data_estratificada2`**.
2. **Feature Selection**: All columns except the 5th one are selected as predictors and stored in **`predictors`**. The 5th column ("BMI") is chosen as the target variable for prediction.
3. **Train-Test Split**: The data is divided into training (**`train.data`**) and testing sets (**`test.data`**) using a 70-30% split ratio.
4. **Initial Linear Regression Model**: An initial linear regression model (**`ins_model`**) is constructed using the training data to predict BMI based on all available predictors. A summary of this model is displayed.
5. **Model Training with Cross-Validation**: A linear regression model is trained using 10-fold cross-validation (**`trainControl`**) on the training data. This involves splitting the data into 10 subsets, training and evaluating the model 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`**).

Analysis:

1. **Normalization Omission**: The code did not include data normalization because it didn't contribute to improved results.
2. **Stratification Avoidance**: Stratification of the sample was intentionally avoided. When applied, it led to abnormal behavior, particularly with variables like "Diabetes_012," in relation to BMI prediction.
3. **Decision Rationale**: The decision not to stratify was made based on the observation that it did not yield meaningful improvements in this specific 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:

```
#### 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)]

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

1. **RMSE (Root Mean Squared Error)**: Model 3 achieves the lowest RMSE (5.9483), signifying the smallest average prediction error among the three models.
2. **R-squared Value**: Model 3 also attains the highest R-squared value (0.1514), indicating that it explains the most variance in the data.
3. **MAE (Mean Absolute Error)**: The lowest MAE is obtained by Model 3 (4.3086), suggesting the smallest absolute prediction errors.

Conclusion: Considering these metrics collectively, Model 3 emerges as the best among the three. It demonstrates the lowest RMSE, the highest R-squared value, and the lowest MAE, collectively suggesting superior predictive performance for BMI prediction.

Linear regression model MentHlth

Model 1

```
### 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

```
###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

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

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

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

1. **RMSE (Root Mean Squared Error)**: Model 3 attains the lowest RMSE (6.5558), indicating the smallest average prediction error among the three models.
2. **R-squared Value**: Model 3 achieves the highest R-squared value (0.2076), signifying that it explains the most variance in the data.
3. **MAE (Mean Absolute Error)**: Model 3 also obtains the lowest MAE (4.0103), suggesting the smallest absolute prediction errors.

Conclusion: Model 3 emerges as the best choice for predicting MentHlth, boasting the lowest RMSE, the highest R-squared value, and the lowest MAE. This performance improvement aligns with the strategy of limiting predictors to those with lower error metrics. In this context, a more focused set of predictors, selected based on their influence on the target variable, leads to a more accurate and effective predictive model.

In summary, for predicting MentHlth, Model 3, with fewer predictors, outperforms others in terms of RMSE, R-squared, and MAE, highlighting the advantage of a focused predictor selection for improved predictive results.

Linear regression model PhysHlth

Model 1

```
#### Linear regression model PhysHlth ##########################################################################

set.seed(10)
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

1. **Model 1**: RMSE of 6.858562 and R-squared of 0.4208921, indicating a good model fit with room for improvement.
2. **Model 2**: 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.
3. **Model 3**: 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.

General Comparison: All models exhibit comparable performance, with Model 1 appearing the most robust in terms of RMSE and R-squared. However, the three models are quite similar overall.

Limiting Predictors Observation: Despite limiting predictors in Models 2 and 3, there's 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. This choice doesn't compromise predictive capability significantly and may help capture more subtle relationships between predictor variables and the target variable (PhysHlth).