Math 4323- Project: Final Report
Group 11
Members :

Kim Nguyen, PID: 1255456
Loi Ha, PID: 1830955
Cuong Phan, PID: 1851430
Thuy Nguyen, PID: 1834037

NATIVE AMERICAN DIABETES DATA STATISTICAL LEARNING

PART 1: INTRODUCTION

The sedentary lifestyle in modern society has led to a high number of Americans diagnosed with diabetes. According to a CDC report in 2017, more than 100 million Americans are now living with diabetes or prediabetes, which can eventually result in serious health complications including premature death, vision loss, heart disease, stroke, kidney failure. For this reason, our group main goal is to explore and identify the main factors causing diabetes.

Our choice of a dataset is the Pima Indian heritage (subgroup of Native Americans).

OVERVIEW VARIABLES:

Source of data: https://www.kaggle.com/uciml/pima-indians-diabetes-database

This dataset is originally from the National Institute of Diabetes and Digestive and Kidney Diseases. The objective of the dataset is to diagnostically predict whether or not a patient has diabetes, based on certain diagnostic measurements included in the dataset. Several constraints were placed on the selection of these instances from a larger database. In particular, all patients here are females at least 21 years old of Pima Indian heritage.

DESCRIPTION OF VARIABLES:

- **Pregnancies (**Number of times pregnant)
- **Glucose (**Plasma glucose concentration a 2 hours in an oral glucose tolerance test)
  BloodPressure (Diastolic blood pressure (mm Hg))
- **SkinThickness (**Triceps skin fold)
- **Insulin (**2-Hour serum insulin (mu U/ml))
- **BMI (**Body mass index (weight in kg/(height in m)^2))
- **DiabetesPedigreeFunction (**Diabetes pedigree function)
- **Age (**Age (years))
- **Outcome (**Class variable (0 or 1) 268 of 768 are 1, the others are 0)

MAIN GOAL:

In the context of this dataset, our main goal is to predict whether or not a patient has diabetes, based on certain diagnostic measurements. Given predictors variables such as pregnancies , glucose, blood pressure, skin thickness, insulin, BMI, diabetes pedigree function, age, we will predict the value of the response variable.

THE QUESTION: Which will be the **outcome** for a patient is whether 1 (has diabetes) or 0 (does not have diabetes) ?

PART 2: METHODOLOGY

Since we have our response variable, we are dealing specifically with a supervised learning problem. We implement SVM and KNN on this dataset and compare their performance.

A. **K-Nearest Neighbors (KNN) analysis**

In the context of this dataset, we have a response variable outcome in binary value (1 or

$$K(\mathbf{x}_i, \mathbf{x}_j) = exp(-\gamma \sum_{k=1}^{p} (x_{ik} - x_{jk})^2), \gamma > 0$$

0) so KNN classifier can be a good model for classification task in supervised learning. KNN classifier proceeds to identify the K points in the training data are closed to x_0 (this set of points is denoted N_0) and then estimate conditional probability for class j as: Pr (Y=j | X = x_0) = {fraction of points in N_0 whose class equals j} or, in proper math notation,

$$\Pr(Y=j \mid X = x_0) = \frac{1}{K} \sum_{i \in N_0} I(yi = j) \ .$$

Then classify the objects to class with highest estimated probability:

$$\hat{C}knn(x0) = j, \text{ if } \hat{p}j(x0) = max\{\hat{p}1(x0),...,\hat{p}J(x0)\}.$$

The smaller the K is the more flexible the method will be.

Advantages of K-NN: it is pretty intuitive and simple, with relatively high accuracy, has no assumptions, can be used for classification and regression task, and has a variety of distance criteria to choose from (Euclidean Distance; Hamming Distance; Manhattan Distance; Minkowski Distance)

Disadvantages of K-NN: it is computationally expensive because the algorithm stores all of the training data, high memory requirement, sensitive to irrelevant features and the scale of data.

B. **Support Vector Machine (SVM) analysis**

We choose the SVM model because the nature of the SVM model fits the goal of prediction, which is the outcome in binary (either 1 or 0).

Advantages of SVM method: Supposing there is a clear distinction between the two classes 1 (diabetic) and 0 (non-diabetic), SVM will work relatively well if there is a clear margin of separation. Even in the case of being non-linearly separable, SVM can still handle data by non-linear kernels such as polynomial and radial kernels. Moreover, the biggest advantage of kernels is computation. With kernels, we only compute K ($\mathbf{x}i$, $\mathbf{x}j$) for all $\binom{n}{2}$ distinct pairs $\mathbf{x}i$, $\mathbf{x}j$ of training observations, in the original p-dimensional space. Kernels are computed without explicitly working in the enlarged feature space, which in many SVM applications can be so large that computations are intractable.

Disadvantages of SVM method: SVM may not perform very well if the dataset has more noise or the target classes are overlapping.

Support Vector Machines: Radial Kernel equation

A kernel is a function that quantifies the similarity of two observations:

Radial kernel has very local behavior: only nearby training observations determine the class label of a test observation.

PART 3: DATA ANALYSIS

Data-preprocessing

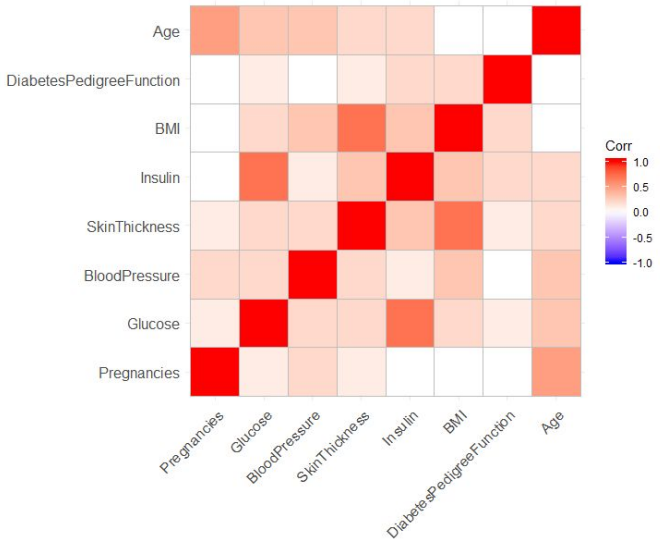
```
# import data
diabetes <- read.csv('C:/Users/wolng/Desktop/math4323/diabetes.csv', header = TRUE)
#imputation using MICE
install.packages('mice')
library(mice)
diabetes[,2:6][diabetes[,2:6]==0] <-NA
#calculate percentage of missing data
missing_percentage_calculated <- function(x){sum(is.na(x))/length(x)*100}
apply(diabetes,2,missing_percentage_calculated)
methods(mice)
processedData <- mice(diabetes[, !names(diabetes) %in% "Outcome"],m=5,maxit=5,meth='norm.predict',seed=500)
#Clean data by imputation
data <- complete(processedData,2)
#Add Outcome back
data$Outcome <- diabetes$Outcome
```

Although there is no n.a (missing data) in this dataset, there are 0 values in glucose, blood pressure, skin thickness, insulin, and BMI columns. We consider 0 values as missing data because it does not make sense for those values to be 0. Instead of omitting the row having 0 values, which can result in potential loss of information, we replace those 0 values with multiple imputation methods (mice in R), which means they will be replaced with chosen/ predicted values from an individual who has similar values on other variables.

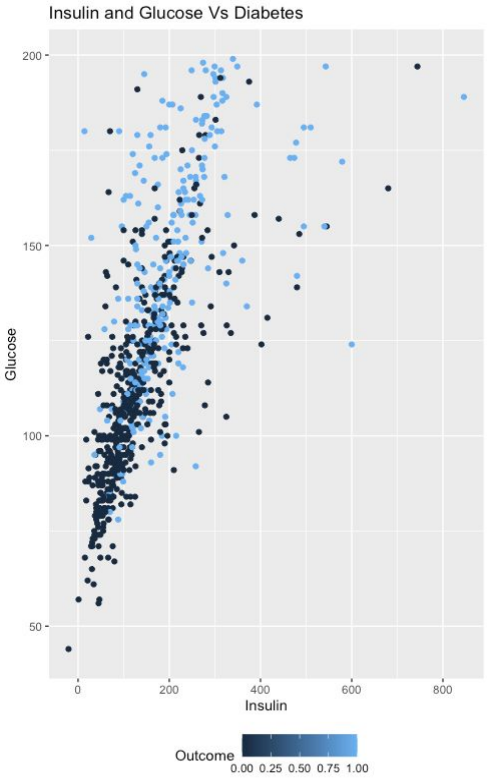
A. PROCESS OF FITTING THE MODEL:

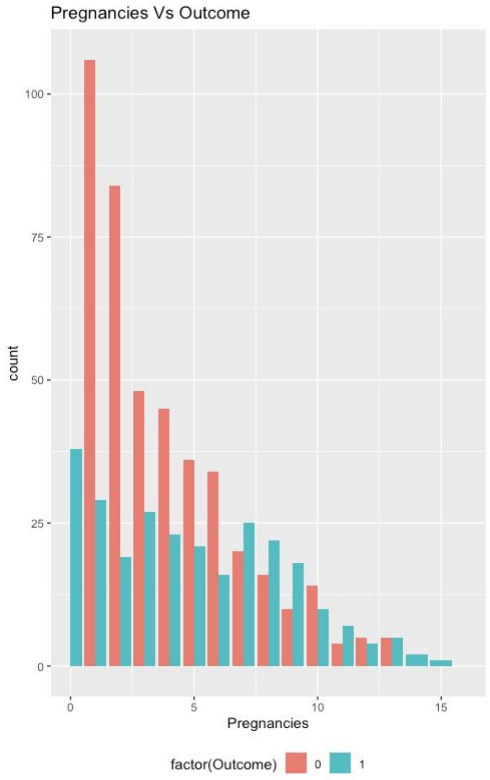
Data correlation matrix

```
# correlation data
set.seed(1)
data_cor <- round(cor(data[1:8]),1)
ggcorrplot(data_cor)
```



➪ No significant case of multicollinearity is observed,there is no strong correlation, strong correlation observed between variables. So, no need to exclude any of them for analysis





Obviously, some factors such as insulin and glucose have correlation with the outcome response variable. The higher values of glucose and insulin are, the higher chance the value of outcome is 1, which means the patient is diabetic. Any other pattern or relationship among those factors will be explored and discussed more in the clustering part.

Data Scaling: One of the most important data transformations we need to apply is the features scaling. Basically most of the machine learning algorithms do not work very well if the features have a different set of values. In our case for example the Age ranges from 20 to 80 years old, while the number of times a patient has been pregnant ranges from 0 to 17. For this reason we need to apply a proper transformation.

```
# scale data
set.seed(1)
data.scale = as.data.frame(scale(data[,-9]))
data.scale$Outcome = data$Outcome
```

Test and Training Set: randomly subdivide your full data set in 80% for training, and 20% for testing

X_test	154 obs. of 8 variables
X_train	614 obs. of 8 variables

table(data.scale$Outcome) =>

0	1
500	268

For the outcome response variable in this dataset, there are 500 non-diabetic and 268 diabetic.

B. SELECT THE OPTIMAL TUNING PARAMETER VALUE

Support Vector Machine - SVM (Cuong Phan, Loi Ha)

Train the training set with three different kernels (linear, polynomial, radial) and compare their performance on unseen data from the test set.

tune() function in R is used to generate the best parameters for each model.

R-code:

```
svm_tune_radial <- tune(svm, Outcome~.,data=train,kernel="radial", ranges =
list(cost=c(0.001,0.1,1,5,10,100),gamma=c(0.5,1,2,3,4,5)))
summary(svm_tune_radial)
svm_tune_linear <- tune(svm, Outcome~.,data=train,kernel="linear", ranges =
list(cost=c(0.001,0.1,1,5,10,100)))
summary(svm_tune_linear)
svm_tune_polynomial=tune(method=svm,Outcome~.,data=train,kernel="polynomial",ranges=li
st(cost=c(0.001, 0.01, 0.1, 1,5,10,100), degree=c(2,3,4)))
summary(svm_tune_polynomial)
```

Best parameters for each kernel report

Kernel	Linear	Polynomial	Radial
Best parameters	cost = 0.1	cost = 0.1, degree = 3	cost = 1, gamma = 0.5

Training the train data set with the best SVM model (radial kernel with cost =1 and gamma = 0.5) yields the following confusion matrix

R-code:

```
pred_train=predict(svm_model,newdata=train)
table(pred_train,train$Outcome)
```

=>

```
pred_train   0   1
         0 368  76
         1  27 143
>
```

```
pred_test = predict(svm_model, newdata = test)
table(pred_test,test$Outcome)
```

=>

```
pred_test  0  1
        0 88 25
        1 17 24
>
```

Based on confusion matrix, the training error is (27 + 76) / (368+143+ 76+ 270) = 0.167752443 and the test error is (17 + 25) / (88+17+24+25) = 0.2727272727

So the accuracy for unseen data in test dataset using SVM is about 72.727%

K- Nearest Neighbor - KNN (Thuy Nguyen, Kim Nguyen)

Select optimal K

We try K = 1 -> 20 and print the error rates for every attempt

```
for (i in 1:20){
  set.seed(1)
  knn.pred <- knn(train=x_train, test=x_test, cl=y_train, k=i)
  print(paste("K = ",i, ": ",(mean(knn.pred!=y_test))))
}
```

```
"K =  1 :  0.305194805194805"
"K =  2 :  0.318181818181818"
"K =  3 :  0.279220779220779"
"K =  4 :  0.285714285714286"
"K =  5 :  0.266233766233766"
"K =  6 :  0.285714285714286"
"K =  7 :  0.246753246753247"
"K =  8 :  0.266233766233766"
"K =  9 :  0.266233766233766"
"K =  10 :  0.279220779220779"
"K =  11 :  0.266233766233766"
"K =  12 :  0.279220779220779"
"K =  13 :  0.298701298701299"
"K =  14 :  0.279220779220779"
"K =  15 :  0.279220779220779"
"K =  16 :  0.279220779220779"
"K =  17 :  0.266233766233766"
"K =  18 :  0.279220779220779"
"K =  19 :  0.246753246753247"
"K =  20 :  0.253246753246753"
```

=> K = 7 gave us the smallest error rate, which is 0.247

```
> set.seed(1)
> knn.pred <- knn(train=x_train, test=x_test, cl=y_train, k=7)
> table(knn.pred, y_test)
        y_test
knn.pred  0  1
       0 85 18
       1 20 31
```

=> accuracy = (85+31)/(85+31+18+20) = 75.3%

C. TEST ERROR PREDICTION RESULT COMPARISON

Compare to SVM (accuracy 72.727%) => KNN is slightly better

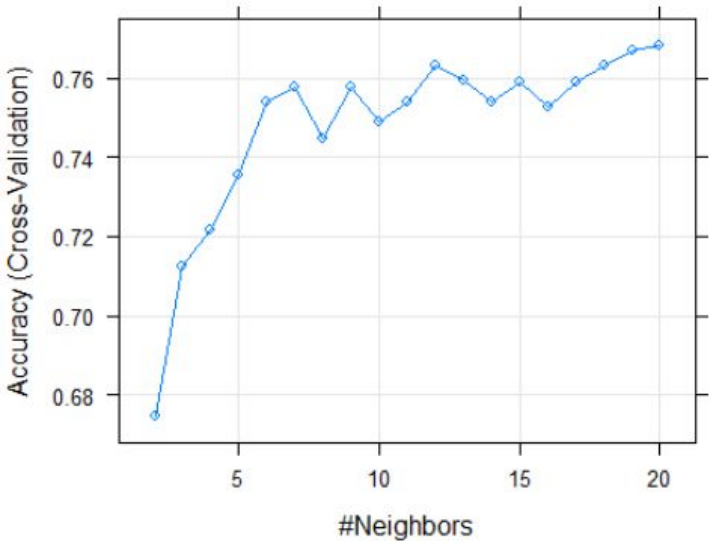
Because KNN is the best model between the two. We fit the whole dataset using KNN.

D. FIT BEST MODEL ON FULL DATA SET

```
grid1 = expand.grid(.k=seq(2,20, by=1))
control = trainControl(method="cv")
set.seed(1)
KNN <- train(Outcome~., data = data.scale, method="knn",
             trControl = control, tuneGrid=grid1)
KNN
```

Summary KNN model applying on full data set
With 10-fold Cross-validation: best accuracy is about 76.83% with k=20

```
k-Nearest Neighbors

768 samples
  8 predictor
  2 classes: '0', '1'

No pre-processing
Resampling: Cross-Validated (10 fold)
Summary of sample sizes: 691, 691, 691, 691, 691, 692, ...
Resampling results across tuning parameters:

  k   Accuracy   Kappa
   2  0.6745557  0.2818584
   3  0.7122864  0.3573114
   4  0.7214457  0.3774086
   5  0.7356972  0.4068404
   6  0.7539645  0.4437552
   7  0.7578947  0.4574420
   8  0.7449248  0.4298550
   9  0.7579289  0.4573484
  10  0.7488038  0.4339033
  11  0.7540670  0.4429341
  12  0.7631579  0.4626049
  13  0.7592276  0.4522764
  14  0.7539474  0.4403562
  15  0.7591763  0.4538061
  16  0.7526658  0.4346776
  17  0.7591763  0.4521191
  18  0.7630895  0.4613720
  19  0.7669686  0.4696731
  20  0.7683014  0.4734552

Accuracy was used to select the optimal model using the largest value.
The final value used for the model was k = 20.
```



Now we use the testset that is the whole dataset that we trained and predicted the Outcome, run it from k=2 to k=20. At k=3 the accuracy is best with 83.98%

```
> for (i in 2:20){
+   set.seed(1)
+   KNN_forloop <- knn(train = data.scale[1:8], test = data.scale[1:8],
+                      cl=data.scale$Outcome, k=i)
+   print(paste("K = ",i, ": ",(mean(KNN_forloop==data.scale$Outcome))))
+ }
[1] "K =  2 :  0.828125"
[1] "K =  3 :  0.83984375"
[1] "K =  4 :  0.830729166666667"
[1] "K =  5 :  0.811197916666667"
[1] "K =  6 :  0.805989583333333"
[1] "K =  7 :  0.8203125"
[1] "K =  8 :  0.805989583333333"
[1] "K =  9 :  0.811197916666667"
[1] "K =  10 :  0.811197916666667"
[1] "K =  11 :  0.803385416666667"
[1] "K =  12 :  0.795572916666667"
[1] "K =  13 :  0.79296875"
[1] "K =  14 :  0.79296875"
[1] "K =  15 :  0.791666666666667"
[1] "K =  16 :  0.787760416666667"
[1] "K =  17 :  0.78515625"
[1] "K =  18 :  0.783854166666667"
[1] "K =  19 :  0.78515625"
[1] "K =  20 :  0.7890625"
```

E. INTERPRETATION OF RESULTS AND CONCLUSION

Compute the confusion matrix (K=3)

```
> KNN.obj <- knn(train = data.scale[1:8], test = data.scale[1:8],
+                  cl=data.scale$Outcome, k=3)
> mean(KNN.obj != data.scale$Outcome)
[1] 0.1601562
> table(KNN.obj, data.scale$Outcome)

KNN.obj   0   1
      0 439  62
      1  61 206
> summary(KNN.obj)
  0   1
501 267
```

For K=3 with KNN fitting the whole scaled dataset, the accuracy is about (439+206)/(439+62+61+206) = 83.98%.

This is the best performance we have achieved in the classification task for this dataset.

PART 4: CONCLUSION

We have developed two supervised learning models, which gives us the results of prediction and accuracy – shows which are the most important factors to have diabetes. Insulin and glucose values are relatively related to each other. The higher values of insulin result in the higher values of glucose and thus, leading to the higher chance of getting diabetes.

One of the main difficulties we have faced during data analysis step is to handle

```
> diabetes[,2:6][diabetes[,2:6]==0] <-NA
> missing_percentage_calculated <- function(x){sum(is.na(x))/length(x)*100}
> apply(diabetes,2,missing_percentage_calculated)
            Pregnancies                 Glucose           BloodPressure           SkinThickness                 Insulin
BMI
              0.0000000               0.6510417               4.5572917              29.5572917              48.6979167
1.4322917
DiabetesPedigreeFunction                     Age                 Outcome
              0.0000000               0.0000000               0.0000000
>
```

missing data (which are 0 values) mostly in skin thickness and insulin columns, which take up to almost 29.557% and 48.6979% of data in the two columns respectively. There are many data-preprocessing methods for handling missing data such as deletion, replace the missing values with mean, median, mode values of each correponding column or imputation. Our approach was to use multiple imputation, which avoids the potential lost information in deletion method and reduce biases in other mean, median, mode methods (which are probably affected by outliers). We have also tried K-means clustering (Appendix section) to discover more about predictors correlation or common pattern among factors. Although the clustering appears to result in two clusters, which possibly

represents two binary values of outcome, the distinction between the two classes are not very clear and the observations are overlapping with each other. We could potentially improve this clustering or prediction result overall by using principal component analysis, which helps us reduce data dimensionality because some predictor factors are actually more important than others.

PART 5: REFERENCE

https://web.maths.unsw.edu.au/~dwarton/missingDataLab.html

PART 6: APPENDIX

K-Means Clustering (KIM NGUYEN)

Based on the data set "Native American Diabetes", we will apply K-means Clustering to find the relationship among 9 variables from 160 different patients. This method helps us to partition the dataset into K distinct and non-overlapping clusters.

- Excluding predictor variable: While the dataset variables already included the diabetes outcome variables, we must exclude it out to get a better training dataset and use the first 8 data set variables as observations. Then from the 8 variables in the dataset we will perform K-Means clustering to group up observations into clustering groups, and we can find the relationship between them that leads to the diabetes output.
- Scaling data variables: The data set includes 8 different, non-related variables. This may lead to instability in learning when machines can tend to weight greater value as higher responsible data. We will scale the dataset to change the values of numeric variables to get every variable to be on a common scale and range, without distorting differences in value.

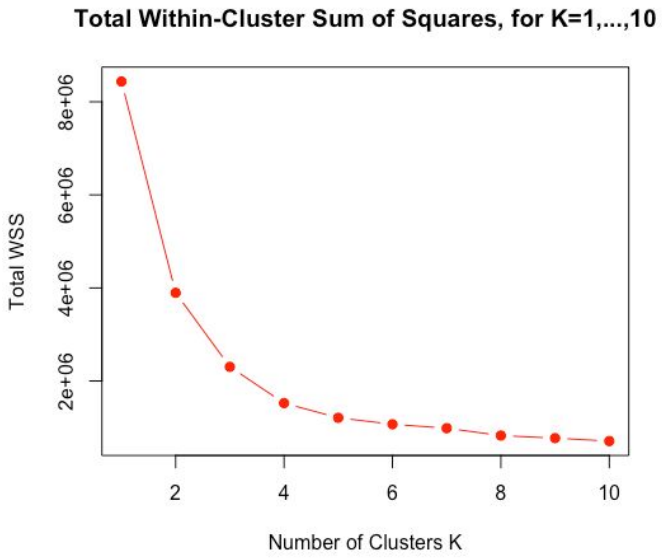
To select the most optimal K value for K-Means clustering, I will run the K-Means algorithm on the dataset for 10 different values of K from 1 to 10, and for each value of K will use 50 random starts. The dataset will exclude the last variables, which is the output for patient diabetes. Also I will use scaling to apply for all of the observations to get the same weight and range for them.

```
# Train scale data set, exclude column 9 - diabetes output
x <- scale(CleanData)
plot(x)
library(factoextra)
kmeans(x, centers=2, iter.max = 30)
k.max <- 10
wss <- numeric(k.max)
for (k in 1:k.max){
  wss[k] <- eclust(x,
                   FUNcluster = "kmeans",
                   k=k,
                   nstart=50)$tot.withinss
}
```

To choose the best K value for K-Means, I will apply two data-based methods on the different results of the Within-Cluster Variation we got after running 10 different values of K in the K-Means algorithm

I. The first data-based method we use is “Elbow methods”. First we plot the total Within-Cluster Variation value correspond to its K value range from 1 to 10

```
plot(wss,
     type="b", pch=19, col=2,
     main = "Total Within-Cluster Sum of Squares, for K=1,...,10",
     xlab = "Number of Clusters K", ylab = "Total WSS")
```



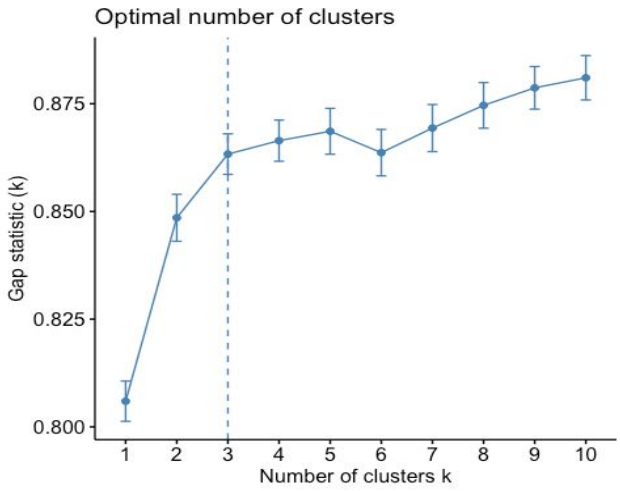
Based on “elbow” method logic, we try to find the smallest K* value such that “The drops in within-cluster variation are the highest up until K* clusters”. We draw a line connecting points (1;WSS1) and (Kmax;WSSKmax). Observe the graph, from the point (K;WSSK) perpendicular connected to the line, we will find **K=2** is the one with the largest distance, corresponding to the optimal K value of clusters.

Using external cluster validation and comparing: Clustering result is K=2 as optimal value, and actual output in data K=2 as 2 value output of diabetes.

Then optimal K=2 value is worked under scale data.

II. The second data-based method we use is “Gap Statistic”, which will compares the curve of log(WSSK) to the original data x, with the curve of log(WSSKUniform) obtained from data xUniform, representing the situation of “no natural groupings of observations”.

```
fviz_nbclust(x, kmeans, nstart = 50, method = "gap_stat", nboot = 50)
```



From the graph, the optimal number of K* of clusters will be the place where the gap between the two curves is largest. The gap statistic is generated from the repeat process of randomly generating uniform data and calculating the gap of each value of K, then calculating the average the gap value across multiple B times.
Observing from the plot of the gap statistic, we observe the optimal value of K is 3.

Using external cluster validation and comparing: clustering result is K=3 as optimal value of gap statistic, with actual output in data K=3 are closed but not same.

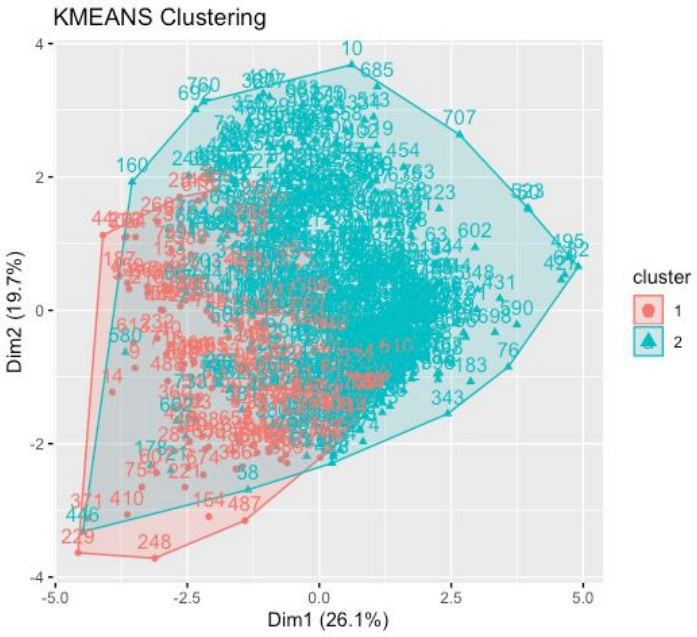
- **Conclusion:** While using the external cluster validation and comparing with the actual output in data (K=2), the applied K-Means algorithm on the dataset to find the Within-Cluster Variation of K value from 1 to 10, also with the use of data-based method "Elbow method" on the plot graph of Within-Cluster Variation, we find the most optimal K value of K-Means clustering is **K =2.**
- **Silhouette coefficient:**

```
ec.obj <- eclust(x,
                 FUNcluster="kmeans", k = 2,
                 nstart = 50)
ec.obj$silinfo

$clus.avg.widths
[1] 0.08825356 0.32053848

$avg.width
[1] 0.2207286
```

- Average silhouette coefficient value = 0.2207286

Fit my best model on a full data set.

```
> diabetes <- read.csv("diabetes.csv")
> x <- diabetes
> eclust (x, FUNcluster = "kmeans", k=2, nstart =50)$tot.withinss
[1] 5142545
```



Based on the plot of K-Means Clustering with K=2 is the optimal K value, we have the result that quite not separates the two clustering well. Then we can conclude that all variable relationships are not strongly related to the output of the diabetes test.