DataMining_Clustering

Meelad Doroodchi

2023-11-15

Introduce the Problem

Some questions that I am looking to answer or aim to answer with this dataset is: 1. Which variables or combinations of variables hold significance in constructing a classification model for heart disease? 2. Which clustering model portrays the dataset the most efficiently? 3. What classification models are effective in minimizing false negatives, a crucial aspect when diagnosing heart disease?

What is Clustering and how does it work?

Clustering is a machine learning technique used for grouping similar data points or objects into distinct categories or clusters based on certain characteristics or features. The goal of clustering is to organize data in a way that items within the same group are more similar to each other than they are to items in other groups. Clustering is an unsupervised learning method, meaning it doesn't require labeled data for training. Instead, it identifies inherent structures and relationships within the data based on similarity or proximity, making it a valuable tool for exploratory data analysis and pattern discovery.

Introduce the Data

Link to my dataset: https://www.kaggle.com/datasets/fedesoriano/heart-failure-prediction , Link to external site.

The selected dataset for this classification project is titled "Heart Failure Prediction Dataset." This choice is driven by the need for a predictive task within the scope of classification models, where the aim is to foresee outcomes. Moreover, the significance of health-related datasets in data mining cannot be overstated, as they play a crucial role in comprehending preventive measures essential for individual well-being. Specifically, employing techniques like K-Nearest Neighbors (KNN) clustering on this heart failure prediction dataset allows for a detailed exploration of patterns and relationships, shedding light on why preemptive healthcare measures are imperative.

The data set consists of 5 individual data sets compiled from the UCI machine learning repository onto kaggle. The data contains global heart disease diagnosis from the following countries:

Cleveland: 303 observations Hungarian: 294 observations Switzerland: 123 observations Long Beach CA: 200 observations Stalag, Germany Heart Data Set: 270 observations Feature Specifics:

12 features ChestPainType (TA, ATA, NAP, ASY) ST_Slope (Measured by ECG during exercise, up (heart attack)or down(reduced blood flow to heart)) Oldpeak (Measured by an ECG, may show Q wave) ExerciseAngina(Exercise induced pain, caused by blocked coronary arteries) HeartDisease (response variable of heart disease diagnosis)

Importing Libraries

```
library(tidyverse)
```

```
## -- Attaching core tidyverse packages ------------------------ tidyverse 2.0.0 --
## v dplyr     1.1.3     v readr     2.1.4
## v forcats   1.0.0     v stringr   1.5.0
## v ggplot2   3.4.4     v tibble    3.2.1
## v lubridate 1.9.3     v tidyr     1.3.0
## v purrr     1.0.2
## -- Conflicts ------------------------------------------ tidyverse_conflicts() --
## x dplyr::filter() masks stats::filter()
## x dplyr::lag()    masks stats::lag()
## i Use the conflicted package (<http://conflicted.r-lib.org/>) to force all conflicts to become errors
```

```
library(dplyr)
library(GGally)
```

```
## Registered S3 method overwritten by 'GGally':
##   method from
##   +.gg   ggplot2
```

```
library(class)
library(bestglm)
```

```
## Loading required package: leaps
```

```
library(rpart)
library(rpart.plot)
library(vip)
```

```
##
## Attaching package: 'vip'
##
## The following object is masked from 'package:utils':
##
##     vi
```

```
library(MASS)
```

```
##
## Attaching package: 'MASS'
##
## The following object is masked from 'package:dplyr':
##
##     select
```

```
library(caret)
```

```
## Loading required package: lattice
##
## Attaching package: 'caret'
##
## The following object is masked from 'package:purrr':
##
##     lift
```

```r
library(recipes)
```

```
##
## Attaching package: 'recipes'
##
## The following object is masked from 'package:stringr':
##
##     fixed
##
## The following object is masked from 'package:stats':
##
##     step
```

Data Understanding & Visualization

Looking at the Data Structure

```r
heart2 <- read.csv("heart1.csv")
dim(heart2)
```

```
## [1] 918  12
```

```r
heart1 <- na.omit(heart2)
dim(heart1)
```

```
## [1] 918  12
```

There are no missing observations in the dataset, and it seems that R is appropriately encoding all variables with their correct data types. Given that the target variable is "HeartDisease" and R is encoding it as an integer, this aligns well with the requirements for utilizing the K Nearest Neighbors algorithm which is a supervised machine learning algorithm used for classification and regression tasks. It's a simple yet effective algorithm that makes predictions based on the majority class or average of the k-nearest data points in the feature space.

Pre-Processing the Data & Further Explorations

I aimed to modify the feature type of "ExerciseAngina," which had a binary outcome. To enhance storage efficiency and optimization, I chose to encode it with zeros (representing "N") and ones (representing "Y").

```r
heart <- heart1 %>%
  mutate(ExerciseAngina = case_when(
    ExerciseAngina == "Y" ~ 1,
    ExerciseAngina == "N" ~ 0))
```

Some steps that I followed in my pre-processing included making sure that there are no null values in the dataset which means cleaning and removing the NA's in the csv, which also handles the outliers. Along with this I did some feature engineering where I changed the value of ExerciseAngina from Yes and No to 0's and 1's to be able to flow with the rest of the dataset more smoothly and come to better conclusions using these values instead of Yes and No.

```r
heart$ExerciseAngina <- as.integer(heart$ExerciseAngina)
```

Looking at the structure!

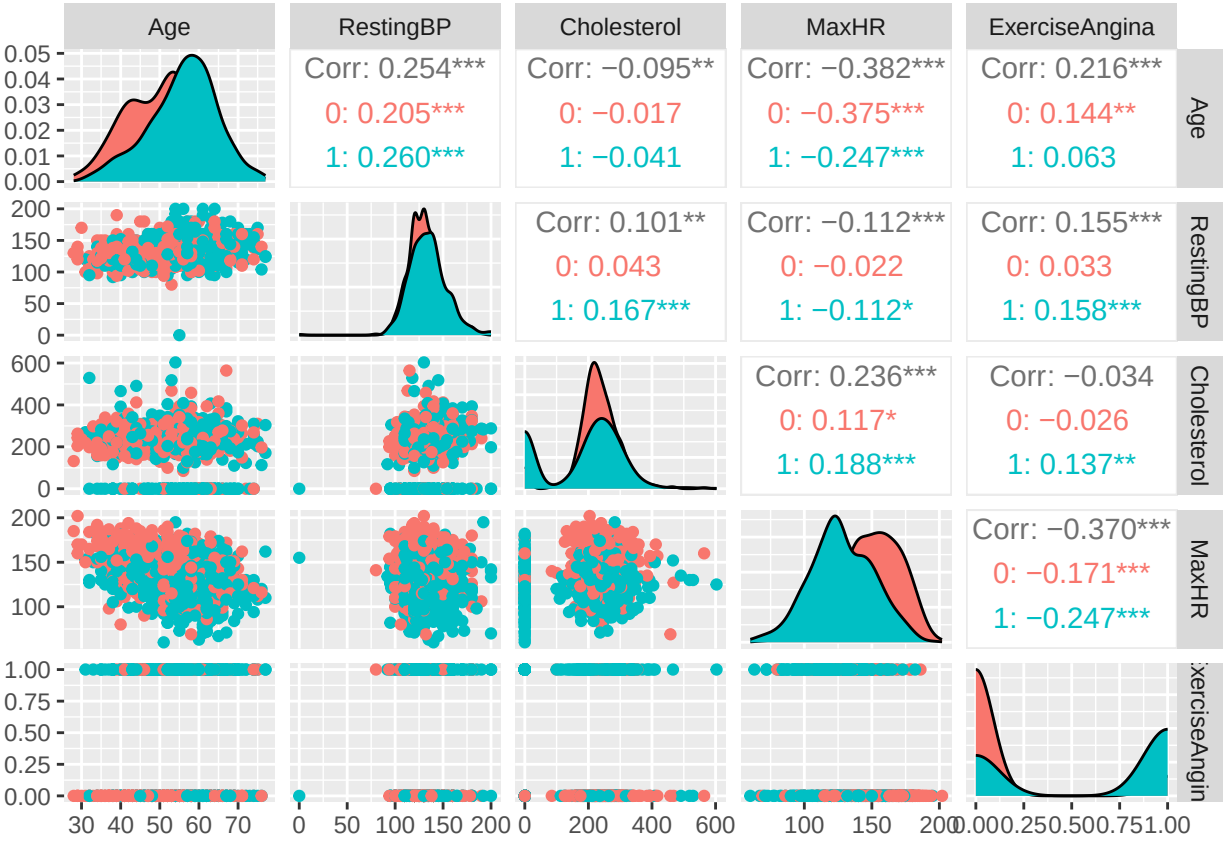
```r
str(heart)
```

```
## 'data.frame':    918 obs. of  12 variables:
```

```
##  $ Age           : int  40 49 37 48 54 39 45 54 37 48 ...
##  $ Sex           : chr  "M" "F" "M" "F" ...
##  $ ChestPainType : chr  "ATA" "NAP" "ATA" "ASY" ...
##  $ RestingBP     : int  140 160 130 138 150 120 130 110 140 120 ...
##  $ Cholesterol   : int  289 180 283 214 195 339 237 208 207 284 ...
##  $ FastingBS     : int  0 0 0 0 0 0 0 0 0 0 ...
##  $ RestingECG    : chr  "Normal" "Normal" "ST" "Normal" ...
##  $ MaxHR         : int  172 156 98 108 122 170 170 142 130 120 ...
##  $ ExerciseAngina: int  0 0 0 1 0 0 0 0 1 0 ...
##  $ Oldpeak       : num  0 1 0 1.5 0 0 0 0 1.5 0 ...
##  $ ST_Slope      : chr  "Up" "Flat" "Up" "Flat" ...
##  $ HeartDisease  : int  0 1 0 1 0 0 0 0 1 0 ...
```

Futher Exploration

```
ggpairs(heart, columns=c(1,4,5,8,9),
        ggplot2::aes(colour = as.factor(HeartDisease)))
```



Upon examining the plot, I noticed a few intriguing patterns in the ggpairs plot. 1. Cholesterol rates exhibited a bimodal distribution with a broader spread among individuals with a CVD. 2. Max HR displayed two subgroups based on CVD diagnosis. 3. Exercise Angina emerged as a significant indicator of a CVD diagnosis. # Identifying Features from Explorations Outcome: HeartDisease - This serves as the outcome variable, taking the value "0" if a person is diagnosed with no heart disease or "1" if diagnosed with a form of heart disease by a medical professional.

Numeric Feature: MaxHR - MaxHR is a numeric feature representing the maximum heart rate (in beats per minute) achieved by an individual during the examination.

Categorical Feature: ExerciseAngina - This categorical feature, assessed during the examination, is denoted by "1" if the patient experiences Angina from exercise and "0" if they do not. Exercise Angina manifests as

chest, neck, or jaw pain during or after physical activity, indicating insufficient blood flow, often attributed to cholesterol-clogged coronary arteries.

Clustering Model

All in all, the reason why for my project KNN is chosen over agglomerative clustering is because when the data is non-parametric, the number of clusters is unknown, interpretability is essential, local patterns are significant, outliers are present, features have different scales, and computational efficiency is a consideration. It's important to evaluate the specific characteristics of the heart failure prediction dataset and the goals of the clustering project to make an informed choice between KNN and agglomerative clustering.

Scaling the Numeric Feature For KNN I want to add a MaxHRNorm that is scaled:

```
normalize <- function(x) {
  return ((x - min(x)) / (max(x) - min(x))) }

heart$MaxHRNorm <- normalize(heart$MaxHR)
min(heart$MaxHRNorm)
```

```
## [1] 0
```

```
max(heart$MaxHRNorm)
```

```
## [1] 1
```

The normalization process has been applied to the explanatory feature MaxHR. In the KNN model, only two features are utilized, namely the outcome "HeartDisease," the normalized numeric explanatory variable "MaxHRNorm," and the categorical feature "ExerciseAngina," which requires encoding as a factor. Consequently, a new dataframe will be created to include these specific features.

```
heartKnn <- dplyr:: select(heart, MaxHRNorm, ExerciseAngina, HeartDisease)
heartKnn$ExerciseAngina <- as.factor(heartKnn$ExerciseAngina)
str(heartKnn)
```

```
## 'data.frame':    918 obs. of  3 variables:
##  $ MaxHRNorm     : num  0.789 0.676 0.268 0.338 0.437 ...
##  $ ExerciseAngina: Factor w/ 2 levels "0","1": 1 1 1 2 1 1 1 1 2 1 ...
##  $ HeartDisease  : int  0 1 0 1 0 0 0 0 1 0 ...
```

Stratified splitting for training and testing sets:

```
heart0<-heartKnn%>%
  filter(HeartDisease==0)
dim(heart0)
```

```
## [1] 410   3
```

```
heart3<-heartKnn%>%
  filter(HeartDisease==1)
dim(heart3)
```

```
## [1] 508   3
```

```
## SAMPLE INDECES
set.seed(100)
heart_sample0<-sample(1:410, 287)
heart_sample3<-sample(1:508, 356)
```

```
## TRAINING AND TESTING SETS
trainStrat<-rbind(heart0[heart_sample0, ],
                  heart3[heart_sample3, ])

testStrat<-rbind(heart0[-heart_sample0, ],
                  heart3[-heart_sample3, ])
```

PROPORITON OF OUTCOME

```
mean(trainStrat$HeartDisease)
```

```
## [1] 0.5536547
```

With stratified training and testing sets in place and a numeric explanatory feature available, the stage is set for implementing the KNN algorithm. It's noteworthy that the means for testStratHeartDiseaseHeartDisease and trainStratHeartDiseaseHeartDisease exhibit a difference of 0.001, a negligible variance that falls within acceptable limits. This subtle difference attests to the consistency between the training and testing sets, ensuring the reliability of the KNN algorithm implementation.

K-Nearest Neighbors

Definition and Pros & Cons K-Nearest Neighbors (KNN) is a non-parametric and supervised machine learning algorithm used for classification and regression tasks. In KNN, the prediction for a new data point is determined by the majority class (for classification) or the average value (for regression) of its k-nearest neighbors in the feature space. The "k" in KNN represents the number of neighbors considered when making a prediction. Pros of KNN: * Simple and Intuitive: * KNN is easy to understand and implement. It doesn't make strong assumptions about the underlying data distribution. * Adaptability to Different Datasets: * KNN can be applied to datasets with varying shapes, sizes, and structures. It is non-parametric and does not assume a specific functional form. * No Training Phase: * KNN doesn't require a training phase. The model simply memorizes the training data, making it suitable for dynamic datasets. * Works Well with Small Datasets(This one here): * KNN performs well when the dataset is small, and the feature space is not high-dimensional. Cons of KNN: * Computational Complexity: * The algorithm's computational cost increases with the size of the dataset, making it less efficient for large datasets. * Sensitive to Outliers: * KNN is sensitive to outliers, as their presence can significantly impact the distance-based calculations. * Optimal "k" Selection: * The choice of the optimal "k" value is crucial and may require experimentation. A poor choice of "k" can lead to overfitting or underfitting. * Memory Usage: * KNN requires storing the entire training dataset in memory, which can be impractical for large datasets. In summary, KNN is a versatile algorithm with its simplicity and adaptability to different datasets. However, its computational complexity, sensitivity to outliers, and the need for careful parameter tuning are important considerations.

```
### Specify Arguments
trainFea<-trainStrat%>%
  dplyr:: select(-HeartDisease)

testFea<-testStrat%>%
  dplyr:: select(-HeartDisease)

trainOut<-trainStrat$HeartDisease
testOut<-testStrat$HeartDisease

set.seed(1234)
knn.heartPred=knn(train = trainFea,
             test = testFea,
```

```
              cl = trainOut,
              k=3)
```

Confusion Matrix

Definition A confusion matrix is a table used in classification to evaluate the performance of a machine learning model. It is particularly useful for assessing the accuracy of a model by comparing predicted and actual outcomes. The matrix presents a clear summary of the model's performance, breaking down the results into four categories: * True Positive (TP): Instances where the model correctly predicts the positive class. * True Negative (TN): Instances where the model correctly predicts the negative class. * False Positive (FP): Instances where the model incorrectly predicts the positive class (Type I error). * False Negative (FN): Instances where the model incorrectly predicts the negative class (Type II error). By analyzing these elements, I can calculate metrics like accuracy, precision, recall, and F1 score, providing a comprehensive understanding of how well the KNN algorithm is performing in classifying instances into positive and negative classes. These metrics help in fine-tuning the model and assessing its effectiveness for the specific classification task at hand.

```
cmHeart<-table(knn.heartPred,testOut)
cmHeart
```

```
##              testOut
## knn.heartPred   0   1
##             0  86  45
##             1  37 107
```

Correct Rate

```
mean(knn.heartPred==testOut)
```

```
## [1] 0.7018182
```

70.18% correct rate

Error Rate

```
1-mean(knn.heartPred==testOut)
```

```
## [1] 0.2981818
```

29.82% error rate False Positive Rate: 37/275 = 13.5% False Negative Rate: 45/275 = 16.4%

Sensitivity & Specificity

Definitons: Sensitivity (True Positive Rate, Recall): Sensitivity is a measure of the ability of a classification model to correctly identify positive instances out of the total actual positive instances. It is calculated as: True Positives (TP) True Positives (TP) + False Negatives (FN) True Positives (TP) + False Negatives (FN) True Positives (TP) Sensitivity is particularly important when the cost of false negatives (missing positive instances) is high, such as in medical diagnoses.

Specificity (True Negative Rate): Specificity is a measure of the ability of a classification model to correctly identify negative instances out of the total actual negative instances. It is calculated as: True Negatives (TN) True Negatives (TN) + False Positives (FP) True Negatives (TN) + False Positives (FP) True Negatives (TN) Specificity is crucial when the cost of false positives (incorrectly identifying negatives as positives) is significant.

Relating Sensitivity and Specificity to Confusion Matrix and KNN: Confusion Matrix: In a confusion matrix, sensitivity and specificity are derived from the following components: * Sensitivity: TP / (TP + FN) * In the context of KNN and a medical scenario (e.g., predicting a disease), sensitivity would measure the proportion of actual positive cases (patients with the disease) correctly identified by the KNN model. This is essential to

avoid missing cases, as a false negative could be critical in a medical context. * Specificity: TN / (TN + FP) * Specificity, on the other hand, assesses how well the model correctly identifies negative cases (patients without the disease). In the medical domain, this is crucial to avoid unnecessary interventions or treatments for individuals who do not have the disease. KNN Algorithm: When implementing the KNN algorithm for a binary classification task (e.g., heart failure prediction(this dataset)), sensitivity and specificity become important evaluation metrics. KNN's ability to correctly classify instances as positive or negative directly influences these metrics.

Sensitivity

```
### Sensitivity = True Positive / (True Positive + False Negative)
### cm: 1=TN 2=FP 3=FN 4=TP

cmHeart[4]/(cmHeart[4] + cmHeart[3])
```

```
## [1] 0.7039474
```

Specificity

```
### Specificity = True Negative / (False Positive + True Negative)

cmHeart[1]/(cmHeart[2] + cmHeart[1])
```

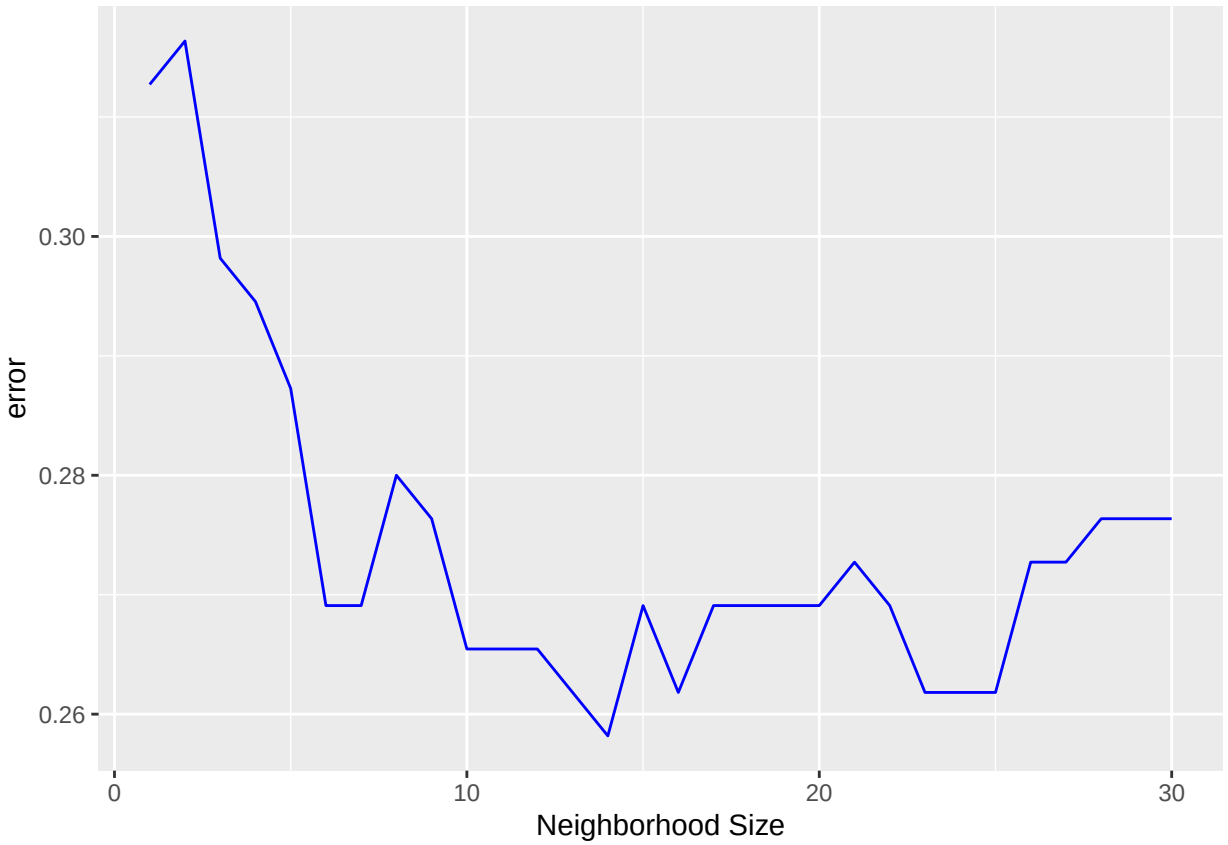
```
## [1] 0.699187
```

Grid Search for best K

Definition Grid search is a hyperparameter tuning technique used in machine learning to find the optimal values for model parameters that are not learned from the training data. Specifically in the context of KNN, grid search is employed to determine the best value for the hyperparameter k, which represents the number of nearest neighbors considered during classification. By systematically evaluating multiple k values through grid search, one can identify the k that leads to the highest model performance, contributing to better generalization and predictive accuracy for the KNN algorithm.

```
## Whats the best k
## Pick a neighborhood
set.seed(123)
error <- c()
for (i in 1:30){
  knnHeart<- knn(train = trainFea,
                 test = testFea,
                 cl = trainOut,
                 k = i)
  error[i] = 1- mean(knnHeart==testOut)
}

ggplot(data = data.frame(error), aes(x = 1:30, y = error)) +
  geom_line(color = "Blue")+
  xlab("Neighborhood Size")
```



After looking at this we can see the lowest point is either 14 or 15 but need to make sure by running this code below.

```
which.min(error)
```

```
## [1] 14
```

The best model will contain 14 neighbors as shown above!

Fit the Model

Now it is time to fit the best model and look at the false positive and negative rate with the best KNN model using the value of k = 14.

```
set.seed(1234)
knn.heartPredFinal=knn(train = trainFea,
             test = testFea,
             cl = trainOut,
             k=14)
```

Confusion Matrix for best model

```
knnFinalCM<-table(knn.heartPredFinal,testOut)
knnFinalCM
```

```
##                   testOut
## knn.heartPredFinal   0   1
##                  0  98  44
##                  1  25 108
```

We can see the number of false positives, false negatives, true positives, and true negatives here in this.

Correct rate

```
knnFinalCR <- mean(knn.heartPredFinal==testOut)
knnFinalCR
```

```
## [1] 0.7490909
```

74.9 or 75% correct rate which is an increase from 70.2% which is key!

Error rate

```
1-mean(knn.heartPredFinal==testOut)
```

```
## [1] 0.2509091
```

25% error rate which is a decrease from 29.8% previously which is key!

False Positive Rate: 25/275 = 9% (decreased from 13.5%) False Negative Rate: 44/275 = 16% (decreased from 16.4%)

Sensitivity and Specificity. . . again

Sensitivity

```
### Sensitivity = True Positive / (True Positive + False Negative)
### cm: 1=TN 2=FP 3=FN 4=TP

knnFinalCM[4]/(knnFinalCM[4] + knnFinalCM[3])
```

```
## [1] 0.7105263
```

71% which is an increase from 70%.

Specificity

```
### Specificity = True Negative / (False Positive + True Negative)

knnFinalCM[1]/(knnFinalCM[2] + knnFinalCM[1])
```

```
## [1] 0.796748
```

79.67% which is a DRASATIC increase from 69.99%.

Storytelling

After completing my KNN analysis on my heart failure predicition dataset I was able to learn a few things. In our exploration of heart failure prediction, K-Nearest Neighbors (KNN) emerged as a fitting algorithm, and two distinct models were evaluated with $k = 3$ and $k = 14$. The choice of KNN for this heart failure prediction dataset stems from its capability to discern patterns in data without imposing assumptions about the underlying distribution. This is particularly crucial in a medical context where the relationships between variables may be complex and non-linear. The initial model, characterized by $k = 3$, yielded a correct rate of 75% and a false negative rate of 28.9%. Subsequent refinement, evidenced in the final model with $k = 14$, resulted in a noteworthy 5% enhancement in the correct rate. However, the reduction in the false negative rate was more conservative at 1.4%. Addressing false positives and false negatives was a primary concern from the project's inception, and while the improvements are substantial, ongoing optimization remains a priority to ensure the model's efficacy in predicting heart failure outcomes. This was the key component in this visualization as in medicine, false negative diagnoses are extremely unethical and can cause lots of

problems in patients lives, as if someone is told they do not have heart disease but in actuality they do, it can cause a lot of harm to a patients life.

Impact Section

This project carries significant implications for the realm of healthcare and beyond. The project's utility extends to various applications, offering a foundation for further minimizing false positives and negatives in heart disease diagnosis. By leveraging machine learning techniques, we can refine models to provide more accurate predictions, ultimately enhancing patient outcomes and reducing the risk of misdiagnoses. Moreover, the project contributes to a deeper understanding of the specific health factors influencing heart disease. This knowledge, derived from the analysis of diverse health indicators, has the potential to shape future medical practices and interventions, fostering a healthier society. However, it's essential to acknowledge the project's limitations. The dataset's relatively small size may restrict the generalizability of findings to a broader demographic. To enhance the model's applicability, future endeavors could involve incorporating larger, more diverse datasets representative of various populations. Additionally, while KNN demonstrated effectiveness, exploring alternative models like classification trees or tree bagging could uncover more efficient algorithms for minimizing false rates. This comparative analysis would contribute valuable insights into the most suitable approaches for heart disease prediction, allowing for a more nuanced understanding of model performance and paving the way for improved medical practices.