Conteúdo

1. VISÃO GERAL E ENTENDENDO O DATASET

O objetivo deste trabalho é aplicar conhecimento de aprendizado não supervisionado utilizando o algoritmo k means e algoritmo do cotovelo (elbow) para comparar o número de clusters geradas por cada um, verificando a classificação não supervisionada e os resultados.

O dataset usado em questão será o mesmo do projeto1 Breast Cancer (Cancer de mama), portanto como foi um dataset já usado, a parte de exploração e entendimento dos dados será mais curta.

1.1 EXPLORAÇÃO INICIAL

Importamos os arquivos .data do dataset, identificamos o arquivo wdbc.data, como sendo o que contia o dataset a ser trabalhado. Identificamos através do número de instâncias no mesmo e o especificado no link: https://archive.ics.uci.edu/ml/datasets/Breast+Cancer+Wisconsin+(Diagnostic) (Links para um site externo.).

Ao perceber que as colunas não possuíam um header, optamos por pesquisar sobre o dataset para dar significado a todos os atributos, para aplicarmos de forma correta o algoritmo KNN e árvore de decisão. Utilizamos outra especificação do mesmo dataset https://www.kaggle.com/uciml/breast-cancer-wisconsin-data, para assim renomear e dar sentido aos atributos que serão trabalhados.

1.2 ENTENDIMENTO DOS DADOS

Avaliando os atributos renomeados juntamente com os dados descritos no site do dataset entendemos que esse é o problema de pacientes que desenvolveram o câncer de mama, seja de forma maligna ou benigna como uma das features já classifica.

Os dados formam um dossiê do câncer observado no paciente, bem como espessura, tonalidade, média de diâmetro e etc.

1.3 ENTENDIMENTO DO PROBLEMA PROPOSTO

Comparar o algoritmo kmeans de aprendizado não supervisionado e o seu resultado em comparação com o algoritmo do cotovelo.

2 . DESENVOLVIMENTO

2.1 EXPLORAÇÃO ESTATÍSTICA DOS DADOS

Inicialmente renomeamos as colunas do dataset usando a mesma abordagem do projeto 1 .

Código

```
data <- read.csv2("wdbc.data", sep =",", na.strings =
c('','NA','na','N/A','n/a','NaN','nan'), header = FALSE,  dec=".",
stringsAsFactors=FALSE)

# Renomeando as colunas com base no indicado no link do dataset
https://www.kaggle.com/uciml/breast-cancer-wisconsin-data
cancers <- data %>%
  rename(
    id = V1,
    diagnosis = V2, # Significado: Câncer de mama M -Maligno, B -
Beligno
    radius_mean = V3, # Significado: Raio medio do câncer
    texture_mean = V4, # Significado: Média de escala de cinza do
câncer
    perimeter_mean = V5, # Significado: Media de perimetro do câncer
    area_mean = V6, # Significado: Média de area do câncer
    smoothness_mean = V7,
    compactness_mean = V8,
    concavity_mean = V9,
    concave_points = V10,
```

```
    symmetry_mean = V11,
    fractal_dimension = V12,
    radius_se = V13,
    texture_se = V14,
    perimeter_se = V15,
    area_se = V16,
    smoothness_se = V17,
    compactness_se = V18,
    concavity_se = V19,
    concave_points_se = V20,
    symmetry_se = V21,
    fractal_dimension_se = V22,
    radius_worst = V23,
    texture_worst = V24,
    perimeter_worst = V25,
    area_worst = V26,
    smoothness_worst = V27,
    compactness_worst = V28,
    concavity_worst = V29,
    concave_points_worst = V30,
    symmetry_worst = V31,
    fractal_dimension_worst = V32
  )

# Entendimento previo do dataset
dim(cancers)
sum(str_count(cancers$diagnosis, "M")) # 212 Maligno
sum(str_count(cancers$diagnosis, "B")) # 357 Benigno
```

Contabilizamos a quantidade de exemplares benigno e maligno no dataset, sendo 212 exemplares malignos e 357 benignos existindo assim um desbalanceamento na classificação.

2.2 LIMPEZA DOS DADOS

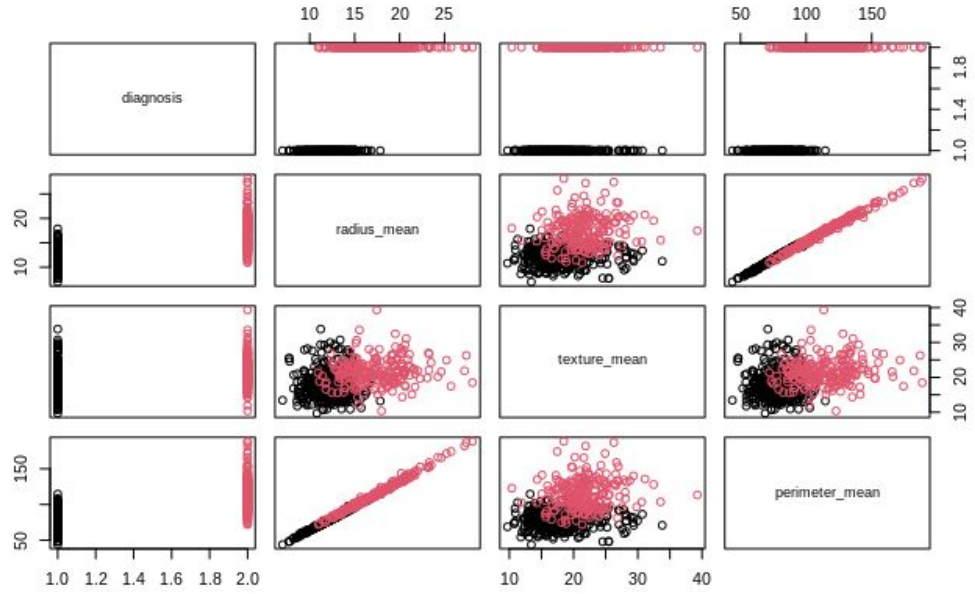
Primeiramente ordenamos o dataset por classificação para termos conhecimento de como comparar com o vetor das classes que será retirado posteriormente. Concluindo a limpeza retiramos os dados de classificação e também a coluna de ID do dataset. Posteriormente realizamos a transformação do

conteúdo de classificação que estava registrado pelas iniciais B - benigno e M - Maligno, de acordo com a ordenação em 1 e 2 respectivamente.

Código

```
#Ordenando o dataset para aplicar o Kmeans
cancers <- cancers[order(cancers$diagnosis),]

data <- cancers[, -1] #Retiro id
data <- data[, -1] #Retiro classificação
classes <-cancers[, 2]

# Comparar obtido com real do dataset

classes <-replace(classes, classes == "B", 1) # Atribuindo benigno
com 1
classes <-replace(classes, classes == "M", 2) # Atribuindo maligno
com 2
classes <- as.numeric(classes)

# plot distribuição das classes considerando primeira features
plot(cancers[, 2:5], col = classes)
```

Com um plot de distribuição dos pontos pelas primeiras features do dataset, por ser um dataset com muitas características, filtrando por tipo de diagnóstico é possível perceber que a muitos dados dos exemplares benigno e maligno se sobrepõem, isso pode acarretar em agrupamentos errados pela similaridade dos dados para alguns exemplares de diferentes classes.



3. Aplicação dos algoritmos

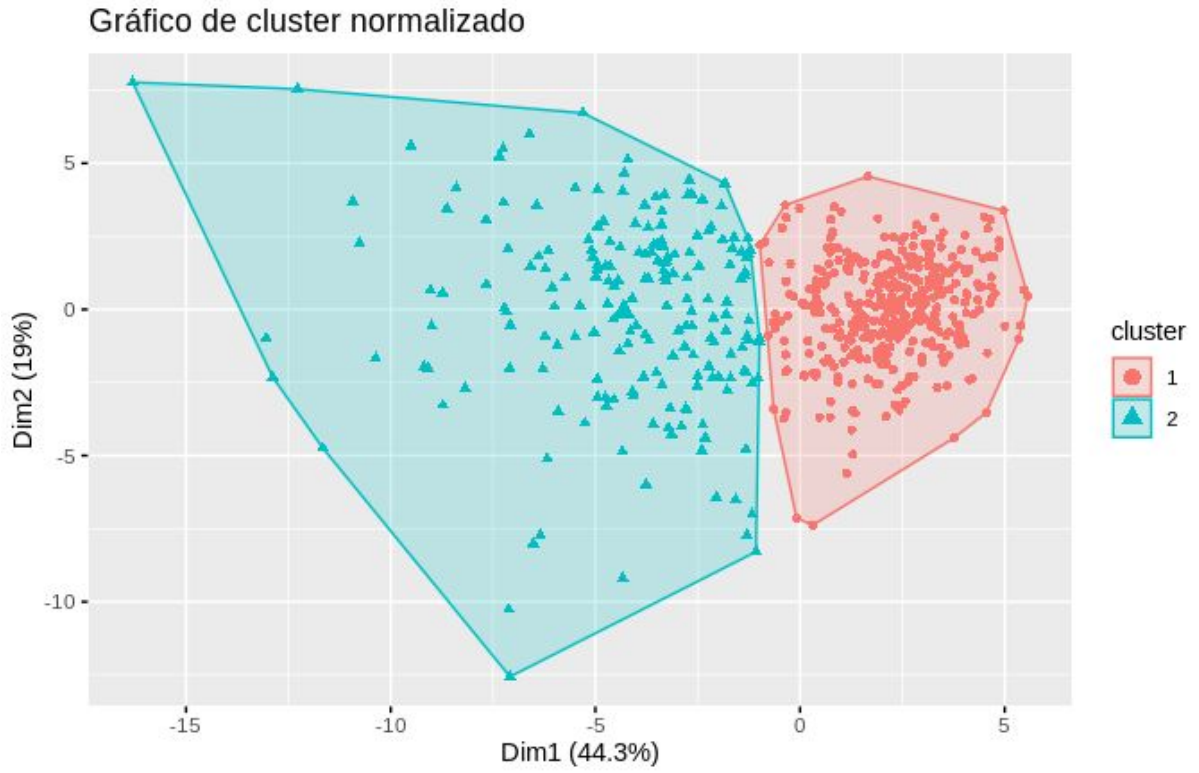
3.1 K MEANS

Como já tínhamos um conhecimento prévio do dataset, usamos o número correto de clusters para o problema, que seria 2. Após obter os resultados, comparamos com a classificação real, verificando a taxa de acerto do algoritmo. Para um melhor desempenho e coerência dos dados ao aplicar o algoritmo foi feito uma normalização através de padronização dos exemplares seguindo a mesma escala, pois ao pesquisarmos e procurarmos entender um pouco mais do algoritmo, vimos que por trabalhar com a minimização as distâncias euclidianas quadradas dentro do cluster, seria interessante aplicar (feature scaling method) para uma padronização das distâncias tanto para exemplares malino quanto benigno.

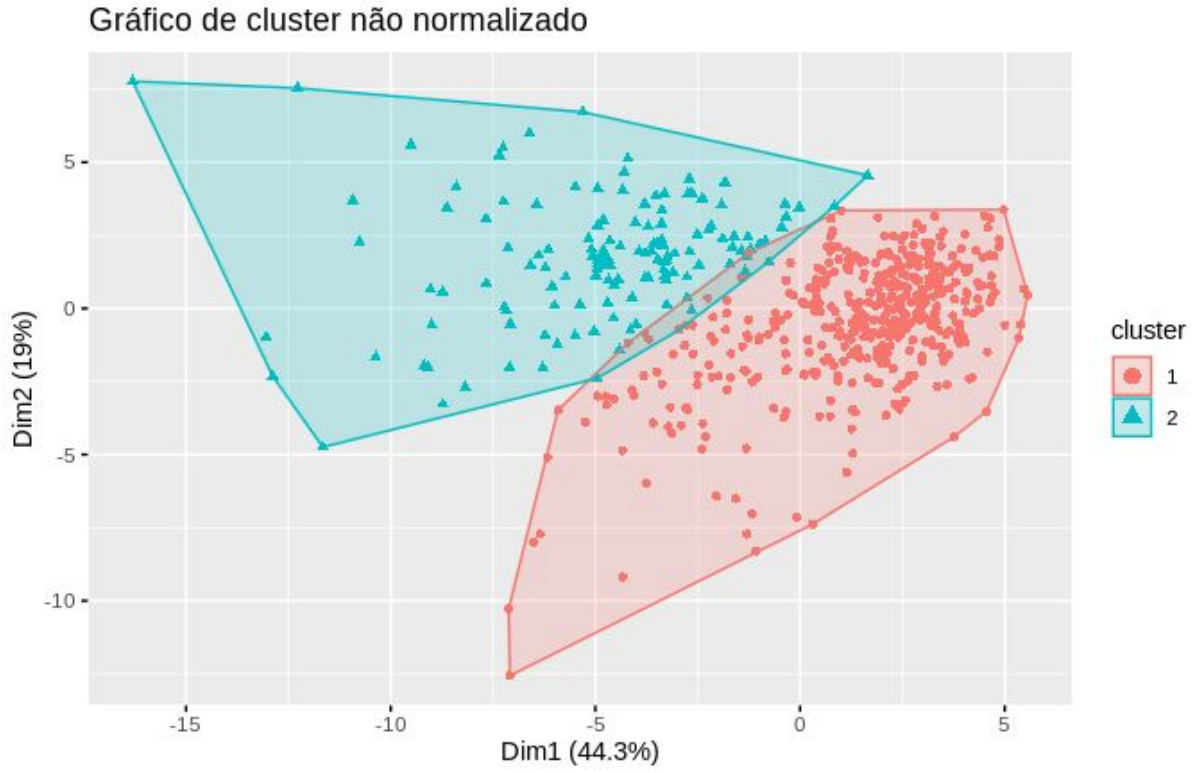
```
# dataset com muitas features, aplicar escala
data_scale<-as.data.frame(scale(data))

cl1 <- kmeans(data_scale, 2)
cl2 <- kmeans(data, 2)

cl$cluster
classes
```

```
compar1 <- classes == cl1$cluster
compar2 <- classes == cl2$cluster

taxaAcerto1 <- (table(compar1)[names(table(compar1)) == TRUE] * 100)
/ length(compar1) # Contando a taxa de acerto: 90,5
taxaAcerto1

taxaAcerto2 <- (table(compar2)[names(table(compar2)) == TRUE] * 100)
/ length(compar2) # Contando a taxa de acerto: 85,4
taxaAcerto2

fviz_cluster(cl1, data = data_scale, geom = "point", main = "Gráfico
de cluster normalizado")
fviz_cluster(cl2, data = data, geom = "point", main = "Gráfico de
cluster não normalizado")
```

Observamos então que a taxa de acerto do que o algoritmo gerou ficou em torno de 90,5%, fazendo uma comparação direta com o seu vetor de classificação real. E sem a normalização por *feature scaling* a taxa de acerto foi menor com 85,4 %, indicando possíveis confusões de agrupamento. Para uma maior visualização dessa porcentagem foi realizado um plot gráfico dos clusters.



Com o conhecimento prévio que o dataset está desbalanceado em relação a classes de diagnóstico com 357 Benigno e 212 Maligno, os dois cluster gerados pelo gráficos demonstra uma proporção de tamanho também diferente correspondente a essa informação.



No gráfico gerado pelo k-means com os dados sem uma escala é possível visualizar uma intersecção entre as regiões dos cluster, indicando uma área de confusão, onde existe a possibilidade de ter perda maior de dados de um clusters agrupado indevidamente em outro.

3.1 ALGORITMO DO COTOVELO

O resumo das nossas pesquisas foram no intuito de entender didaticamente o uso do algoritmo do cotovelo.

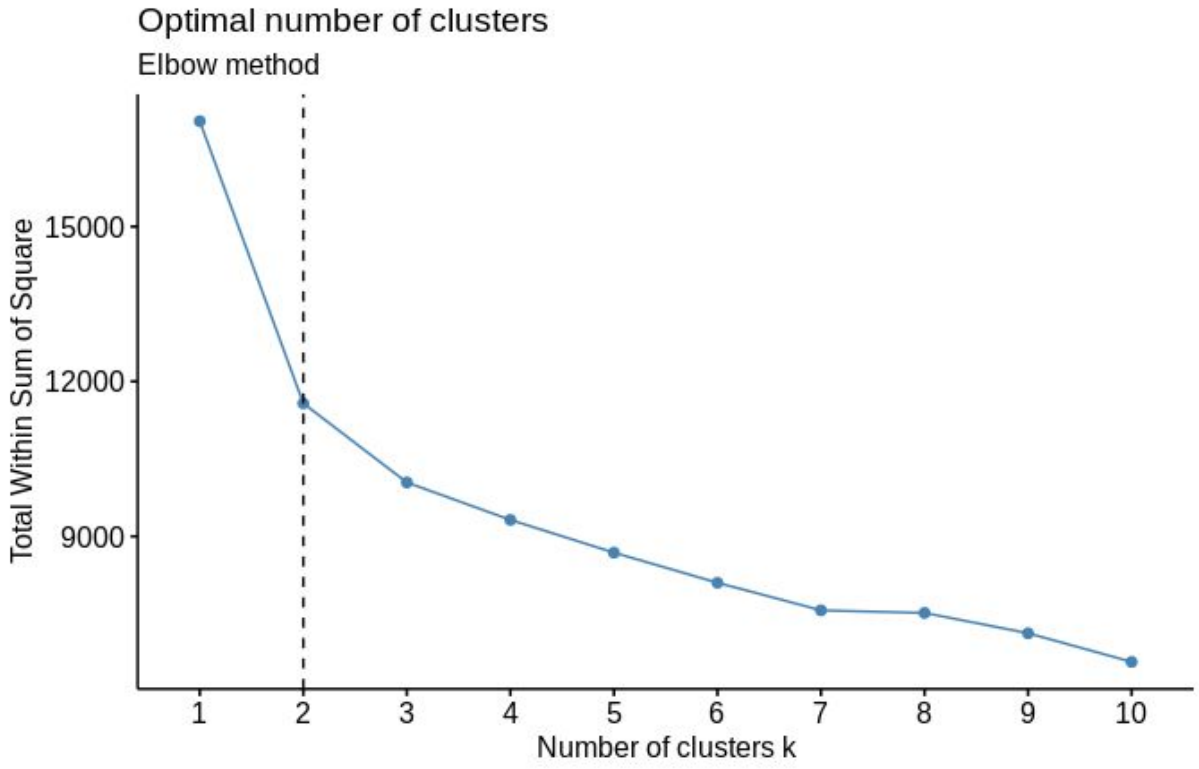
Usando as seguintes fontes:

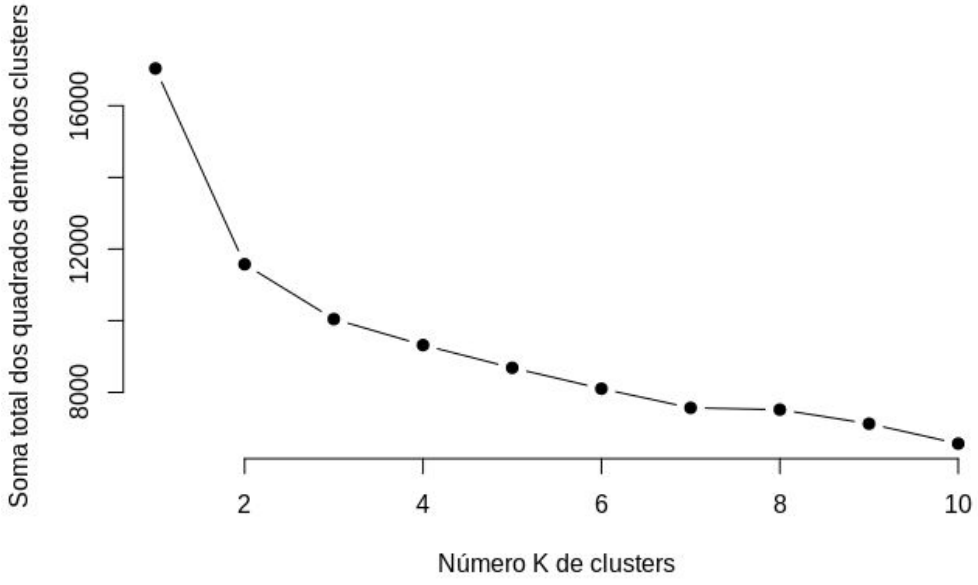
- https://www.datanovia.com/en/lessons/determining-the-optimal-number-of-clusters-3-must-know-methods/
- https://afit-r.github.io/kmeans_clustering

Concluímos que se tratava de aplicar analisar estatisticamente um número ótimo de clusters a partir de um dataset e observar graficamente a formação de um cotovelo em relação ao número ótimo de clusters que deveria ser aplicado ao problema.

Uma das dificuldades de se trabalhar com o algoritmo K-means é encontrar o número de cluster correto para sua aplicação, desta forma o método cotovelo é um método entre outros que auxiliam nessa escolha de melhor número de cluster, a partir de iteração rodando o algoritmo k-means para uma série de K's. Quanto menor o valor dessa soma total dos quadrados, melhor o número de cluster, mas isso pode ser perigoso, pois sempre vai decair a inércia em relação a quanto maior o número de cluster, pois os agrupamentos ficam menores, no entanto podem não ser efetivos com baixa observações contidas nele. Por isso ter conhecimento prévio do problema juntamente com o método é essencial para definir o número de cluster ideal para o algoritmo k-means.

Aplicamos o exemplo usando a função fviz_nbclust, onde a mesma recebe o dataset, a função de K means e faz uma soma de quadrados aplicado sobre um número de clusters especificados. Aplicando a função com os dados em escala padronizada obtivemos o seguinte gráfico:



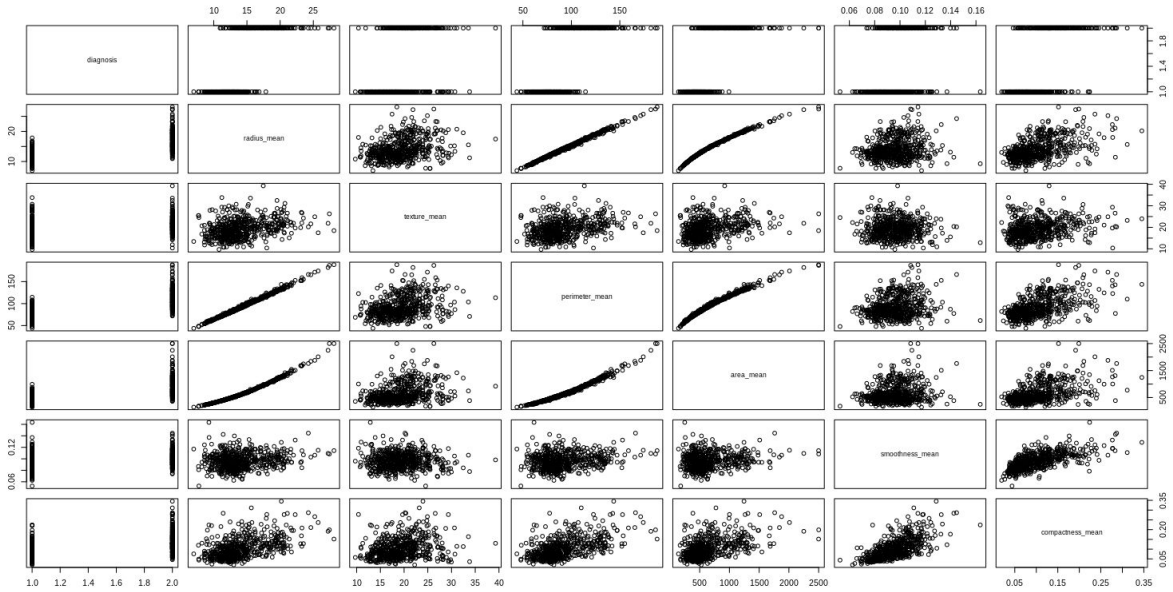


O gráfico foi montado com os seguintes dados gerados pela função:

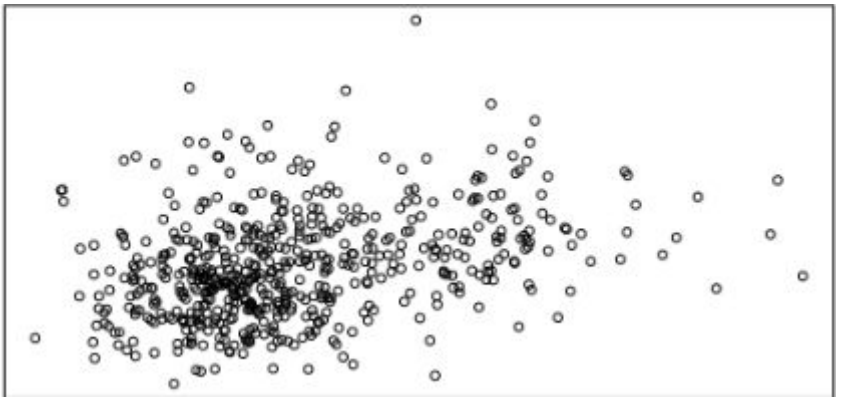
clusters	y
1	17040.000
2	11575.148
3	10044.115
4	9318.804
5	8684.297
6	8101.415
7	7567.399
8	7517.206
9	7123.642
10	6568.558

Há uma queda brusca entre a quantidade de 1 cluster e 2 clusters, desta forma entende -se que a partir de 2 clusters o k-means será mais assertivo, e com o conhecimento prévio do problema de câncer de mama, sabemos que são dois tipos possíveis, ou seja, o ideal é 2 agrupamentos. Isso também é confirmado pelo gráfico de pontos por cada feature, filtrado por tipo diagnóstico na exploração inicial do dataset.

Ao comparar o resultado do algoritmo cotovelo que indicou mais de 2 cluster como ótimo através do gráfico de pontos do dataset original, percebemos que a indicação dos cluster fazem sentido, pois o dataset de câncer é um conjunto de dados com valores de médias pequenas e que se assemelham muito em uma comparação de duas dimensões entre features, desta forma depois de 1 k cluster ,poderia existir possíveis outros agrupamentos por ser dados próximos e que se misturam muitas vezes.



Plot com a aproximação para maior visualização:



Código

```
# Metodo cotovelo já implementado por fviz_nbclust
n_clust <- fviz_nbclust(data_scale, kmeans, method = "wss") +
  geom_vline(xintercept = 2, linetype = 2)+
  labs(subtitle = "Elbow method")
n_clust

n_clust_list <- n_clust$data
n_clust_list

# plot para visualizar distribuição das classes de forma geral
plot(cancers[, 2:8])
```

4. CONCLUSÃO

Aprendizado não supervisionado aplicando o k means se mostrou uma taxa de acerto pouco menor que k-nn e árvore de decisão (aprendizado supervisionado) e se aproximando mais quando padronizado na escala. A proximidade e agrupamento por similaridade dos exemplares em relação as features se mostrou uma análise efetiva para o problema proposto. Aplicando os algoritmos percebemos que com o agrupamento real do problema 2 clusters obtivemos uma taxa de acerto de 90,5%. Para achar um número ótimo de clusters a ser usado no k means, o algoritmo do cotovelo mostra que a partir de 2 cluster é mais efetivo para agrupamento, no entanto ele tende a indicar melhores valores se agrupado com mais K do que 2 cluster, ou seja, a medida que vai aumentando o número de cluster o agrupamento entre as observações são mais próximas e assertivas. Com o

conhecimento prévio do dataset percebemos que a tendência dada pelo algoritmo cotovelo pode confundir o analista sem uma pesquisa ou conhecimento da problemática, por isso deve se atentar as quedas bruscas no total de somas dos quadrados entre as dimensões K, que visualizamos graficamente.

Entendemos que para resolução de problemas não supervisionados o k means se mostrou muito efetivo, e o algoritmo do cotovelo se mostrou um complemento ao k means, onde atua como uma confirmação ou indicação de um possível 'k' ótimo, assim a utilização do método de cotovelo em trabalhos de dataset não supervisionado em situações que necessita de agrupamento de dados, demonstrou -se uma ferramenta muito útil a algoritmo k -means que tem esse ponto fraco de não saber o número prévio ideal de cluster.

5. CÓDIGO

Código

```
library(tidyr)
library(dplyr)
library(ggplot2)
library(dslabs)
library(rpart.plot)
library(tidyverse)  # data manipulation
# install.packages("tidyverse")
library(cluster)    # clustering algorithms
library(factoextra)
library(FactoMineR)
```

```r
data <- read.csv2("wdbc.data", sep =",", na.strings = c('','NA','na','N/A','n/a','NaN','nan'), header = FALSE, dec=".", stringsAsFactors=FALSE)

# Renomeando as colunas com base no indicado no link do dataset https://www.kaggle.com/uciml/breast-cancer-wisconsin-data
cancers <- data %>%
 rename(
   id = V1,
    diagnosis = V2, # Significado: Câncer de mama M -Maligno, B -Beligno
   radius_mean = V3, # Significado: Raio medio do câncer
   texture_mean = V4, # Significado: Média de escala de cinza do câncer
   perimeter_mean = V5, # Significado: Media de perimetro do câncer
   area_mean = V6, # Significado: Média de area do câncer
   smoothness_mean = V7,
   compactness_mean = V8,
   concavity_mean = V9,
   concave_points = V10,
   symmetry_mean = V11,
   fractal_dimension = V12,
   radius_se = V13,
   texture_se = V14,
   perimeter_se = V15,
   area_se = V16,
   smoothness_se = V17,
   compactness_se = V18,
   concavity_se = V19,
   concave_points_se = V20,
   symmetry_se = V21,
   fractal_dimension_se = V22,
   radius_worst = V23,
   texture_worst = V24,
   perimeter_worst = V25,
   area_worst = V26,
   smoothness_worst = V27,
   compactness_worst = V28,
   concavity_worst = V29,
   concave_points_worst = V30,
   symmetry_worst = V31,
   fractal_dimension_worst = V32
 )
```

```r
# Entendimento previo do dataset
dim(cancers)
sum(str_count(cancers$diagnosis, "M")) # 212 Maligno
sum(str_count(cancers$diagnosis, "B")) # 357 Benigno


# Ordenando o dataset para aplicar o Kmeans
cancers <- cancers[order(cancers$diagnosis),] # ordena primeiro benigno
depois maligno

data <- cancers[, -1] # Retiro id
data <- data[, -1] # Retiro classificação

# dataset com muitas features, aplicar escala
data_scale<-as.data.frame(scale(data))

classes <-cancers[, 2]

# Comparar obtido com real do dataset transformando "B" e "M" em dados
numérico
classes <-replace(classes, classes == "B", 1) # Atribuindo benigno com
1
classes <-replace(classes, classes == "M", 2) # Atribuindo maligno com
2
classes <- as.numeric(classes)

# plot para visualizar distribuição das classes considerando primieira
features do dataset
plot(cancers[, 2:5], col = classes)

cl1 <- kmeans(data_scale, 2)
cl2 <- kmeans(data, 2)

cl$cluster
classes

compar1 <- classes == cl1$cluster
compar2 <- classes == cl2$cluster

taxaAcerto1 <- (table(compar1)[names(table(compar1)) == TRUE] * 100) /
length(compar1) # Contando a taxa de acerto: 14,5
taxaAcerto1
```

```r
taxaAcerto2 <- (table(compar2)[names(table(compar2)) == TRUE] * 100) /
length(compar2) # Contando a taxa de acerto: 14,5
taxaAcerto2

fviz_cluster(cl1, data = data_scale, geom = "point", main = "Gráfico de
cluster normalizado")
fviz_cluster(cl2, data = data, geom = "point", main = "Gráfico de
cluster não normalizado")

# -------------- Algoritmo do cotovelo -------------------------------

# função para calcular a soma total do quadrado dentro do cluster
# total within-cluster (tot.withinss)
wss <- function(k) {kmeans(data_scale, k)$tot.withinss}

# Calcule e plote wss para k = 1 a k = 10
k.values <- 1:10

# extrair wss para 2-15 clusters
wss_values <- map_dbl(k.values, wss)

plot(k.values, wss_values,
    type="b", pch = 19, frame = FALSE,
    xlab="Número K de clusters",
    ylab="Soma total dos quadrados dentro dos clusters")

# Metodo cotovelo já implementado por fviz_nbclust
n_clust <- fviz_nbclust(data_scale, kmeans, method = "wss") +
 geom_vline(xintercept = 2, linetype = 2)+
 labs(subtitle = "Elbow method")

n_clust
n_clust_list <- n_clust$data
n_clust_list
```

6. REFERÊNCIAS

---

https://www.datanovia.com/en/lessons/determining-the-optimal-number-of-clusters-3-must-know-methods/ - 08/11/2020: Visitado para pesquisa e entendimento do algoritmo do cotovelo.

https://afit-r.github.io/kmeans_clustering - 08/11/2020: Visitado para pesquisa e entendimento do algoritmo do cotovelo.

https://www.rdocumentation.org/packages/factoextra/versions/1.0.7/topics/fviz_nbclust - 11/11/2020: Visitado para aplicação do algoritmo do cotovelo.