Sardar Patel Institute of Technology, Mumbai

Department of Electronics and Telecommunication Engineering

B.E. Sem-VII- PE-IV (2024-2025)

IT 24 - AI in Healthcare

Experiment 6: Healthcare data analysis in unsupervised learning (K Means Clustering) using R Programming

Name: Adwait Purao **Date:** **14**/10/2024

Objective:

To learn the basics of R programming and RStudio IDE and apply it in healthcare dataset in un-supervised learning algorithm (K Means Clustering), Interpret and visualize Clustering results.

Dataset: *https://www.kaggle.com/datasets/iamhungundji/covid19-symptoms-checker*

The dataset is designed to help identify individuals with coronavirus disease based on defined symptoms aligned with WHO guidelines. It features seven key variables: Country, Age, Symptoms (Fever, Tiredness, Difficulty in breathing, Dry cough, Sore throat), Other Symptoms (Pains, Nasal Congestion, Runny Nose, Diarrhea, Other), Severity (Mild, Moderate, Severe), and Contact with COVID-19 patients. A total of 316,800 combinations of these categorical variables are generated.

There are two CSV files: Raw-Data, which lists all possible variable combinations, and Cleaned-Data, which contains the processed combinations for analysis, including dummy variables. Potential applications of this data include developing a chatbot, and conducting supervised learning (classification) and unsupervised learning (clustering). Future plans involve incorporating more WHO guidelines for expanded data and exploring reinforcement learning techniques.

We are using Cleaned data file as covid_symptoms.csv.

Theory:

K-means Clustering in Healthcare AI/ML

K-means clustering is an unsupervised learning algorithm commonly used in healthcare for various applications.

Basic Concept

K-means partitions n observations into k clusters, where each observation belongs to the cluster with the nearest mean (centroid).

Algorithm Steps

1. Choose k initial centroids
2. Assign each data point to the nearest centroid
3. Recalculate centroids based on assigned points
4. Repeat steps 2-3 until convergence

Applications in Healthcare

- Patient Segmentation
- Disease Subtyping
- Medical Image Analysis
- Drug Discovery

Example: Patient Segmentation

Input: Patient data (age, BMI, blood pressure, etc.)

Output: k patient clusters

1. Initialize k centroids randomly
2. Assign patients to nearest centroid based on their features
3. Update centroids as mean of assigned patients
4. Repeat 2-3 until stable

Advantages

- Simplicity and scalability
- Useful for discovering hidden patterns in medical data

Limitations

- Requires predefined k
- Sensitive to initial centroid selection
- Assumes spherical clusters

Experiment (Steps):

1. Install R and RStudio.
2. Install packages tidyverse, ggplot2, and dplyr.
3. Import a healthcare dataset
4. Load the Dataset
5. Data pre-processing
6. Choose the Number of Clusters(Using Elbow Method
7. Apply K-Means Clustering
8. Segment patients into risk groups
9. Interpret and visualize the Clusters

Code & Output:

```
install.packages(c("tidyverse", "ggplot2", "dplyr", "factoextra"))
library(tidyverse)
library(ggplot2)
library(dplyr)
library(factoextra)
```

```
Installing packages into ‘/usr/local/lib/R/site-library’
(as ‘lib’ is unspecified)

── Attaching core tidyverse packages ──────────────────────── tidyverse 2.0.0
──
✔ dplyr     1.1.4     ✔ readr     2.1.5
✔ forcats   1.0.0     ✔ stringr   1.5.1
✔ ggplot2   3.5.1     ✔ tibble    3.2.1
✔ lubridate 1.9.3     ✔ tidyr     1.3.1
✔ purrr     1.0.2
── Conflicts ────────────────────────────────────────── tidyverse_conflicts()
──
✖ dplyr::filter() masks stats::filter()
✖ dplyr::lag()    masks stats::lag()
ℹ Use the conflicted package (<http://conflicted.r-lib.org/>) to force all con
flicts to become errors
Welcome! Want to learn more? See two factoextra-related books at https://goo.g
l/ve3WBa
```

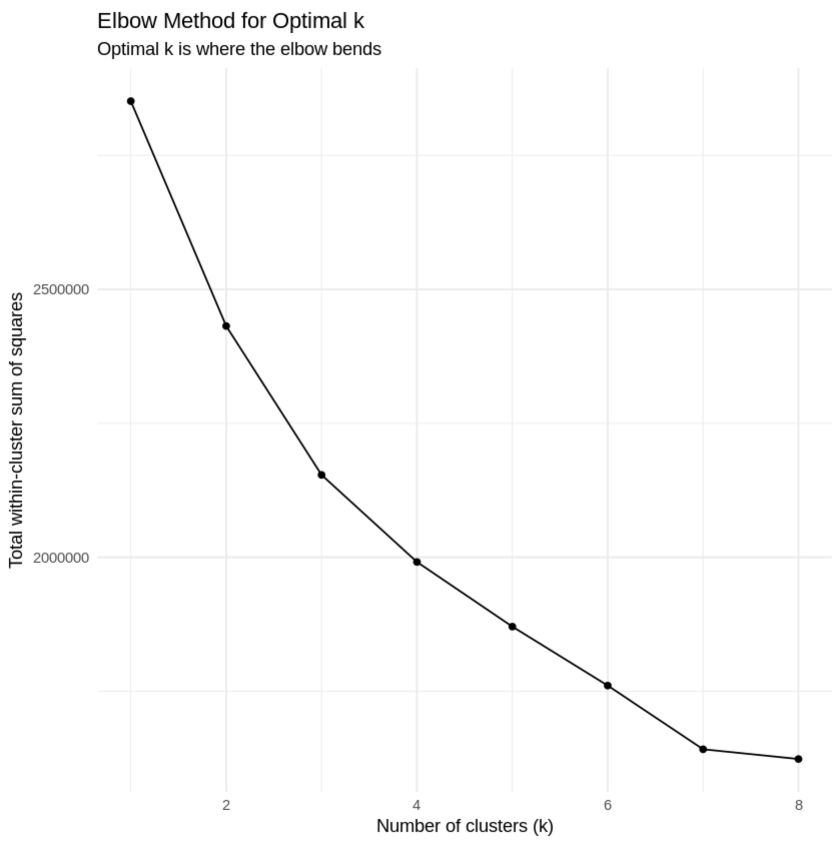
```
covid_data <- read.csv("covid_symptoms.csv")
head(covid_data)
```

```
  Fever Tiredness Dry.Cough Difficulty.in.Breathing Sore.Throat None_Sympton
1 1     1         1         1                       1           0
2 1     1         1         1                       1           0
3 1     1         1         1                       1           0
4 1     1         1         1                       1           0
```

```
5 1     1         1         1                       1          0
6 1     1         1         1                       1          0
  Pains Nasal.Congestion Runny.Nose Diarrhea ⋯ Gender_Male Gender_Transgender
1 1     1                1          1        ⋯ 1           0
2 1     1                1          1        ⋯ 1           0
3 1     1                1          1        ⋯ 1           0
4 1     1                1          1        ⋯ 1           0
5 1     1                1          1        ⋯ 1           0
6 1     1                1          1        ⋯ 1           0
  Severity_Mild Severity_Moderate Severity_None Severity_Severe
1 1             0                 0             0
2 1             0                 0             0
3 1             0                 0             0
4 0             1                 0             0
5 0             1                 0             0
6 0             1                 0             0
  Contact_Dont.Know Contact_No Contact_Yes Country
1 0                 0          1           China
2 0                 1          0           China
3 1                 0          0           China
4 0                 0          1           China
5 0                 1          0           China
6 1                 0          0           China
```

```r
# Select relevant features for clustering
features <- covid_data %>%
  select(Fever, Tiredness, `Dry.Cough`, `Difficulty.in.Breathing`,
         `Sore.Throat`, Pains, `Nasal.Congestion`, `Runny.Nose`, Diarrhea)

# Normalize the data
features_normalized <- scale(features)

library(parallel)
library(factoextra)

# Function to compute WSS for a given k
compute_wss <- function(k) {
  kmeans(features_normalized, centers = k, nstart = 25, iter.max = 100, algorithm = "
Lloyd")$tot.withinss
}

# Compute WSS for k = 1 to 8 in parallel
k_range <- 1:8
num_cores <- detectCores() - 1  # Use all cores except one
wss <- mclapply(k_range, compute_wss, mc.cores = num_cores)

# Convert the result to a numeric vector
wss <- unlist(wss)

# Create a data frame for ggplot
elbow_data <- data.frame(k = k_range, wss = wss)
```

```r
# Plot the elbow curve using ggplot2
elbow_plot <- ggplot(elbow_data, aes(x = k, y = wss)) +
  geom_line() +
  geom_point() +
  labs(x = "Number of clusters (k)",
       y = "Total within-cluster sum of squares",
       title = "Elbow Method for Optimal k",
       subtitle = "Optimal k is where the elbow bends") +
  theme_minimal()

print(elbow_plot)
```



Clearly k should be 3 here

```r
# Apply K-Means Clustering
set.seed(123)  # for reproducibility
k <- 3
kmeans_result <- kmeans(features_normalized, centers = k, nstart = 25, iter.max = 100
, algorithm = "Lloyd")

# Add cluster assignments to the original dataset
covid_data$Cluster <- as.factor(kmeans_result$cluster)

# Segment patients into risk groups
cluster_means <- features_normalized %>%
  as.data.frame() %>%
  mutate(Cluster = kmeans_result$cluster) %>%
  group_by(Cluster) %>%
  summarise(across(everything(), mean)) %>%
  ungroup()
```

```
# Assign risk levels based on symptom severity
risk_scores <- rowSums(cluster_means[,-1])
risk_levels <- c("Low", "Moderate", "High")  # Adjust risk levels to match k=3
cluster_risk <- data.frame(Cluster = as.factor(1:k),  # Convert Cluster to factor
                           Risk = risk_levels[rank(risk_scores)])
```

```
# Join the data
covid_data <- covid_data %>%
  left_join(cluster_risk, by = "Cluster")
```

```
# Quick summary of risk distribution
risk_summary <- table(covid_data$Risk)
print(risk_summary)
```

```
    High      Low Moderate 
   79200   138600    99000 
```

```
risk_summary_prop <- prop.table(table(covid_data$Risk))
print(risk_summary_prop)
```

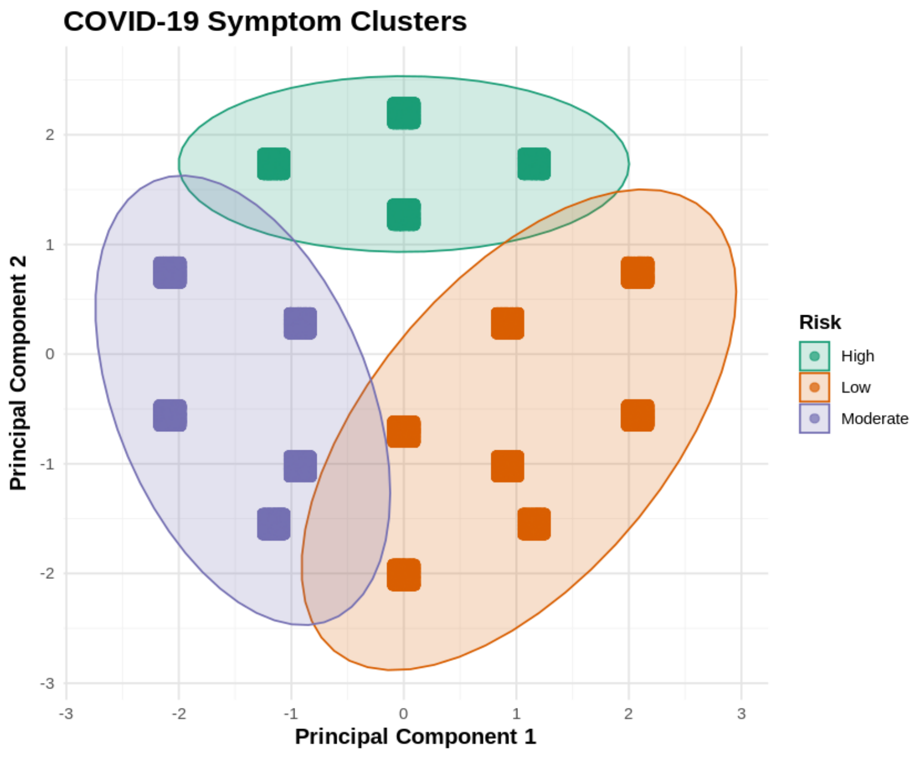
```
    High      Low Moderate 
  0.2500   0.4375   0.3125 
```

```
install.packages(c("ggforce"))
```

```
Installing package into ‘/usr/local/lib/R/site-library’
(as ‘lib’ is unspecified)

also installing the dependencies ‘tweenr’, ‘polyclip’
```

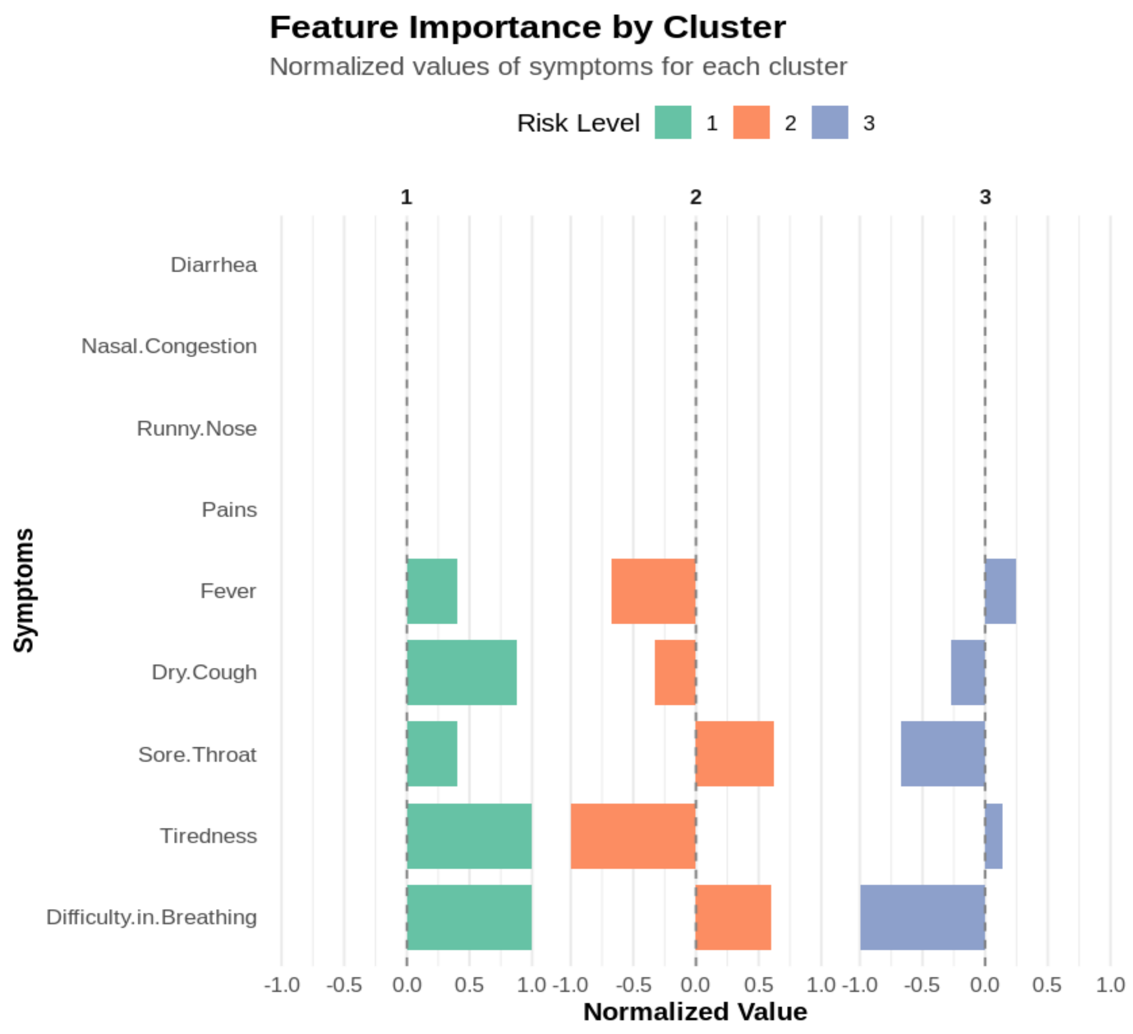
```
library(ggrepel)
library(ggforce)

# Visualize clusters using PCA
pca_result <- prcomp(features_normalized, center = FALSE, scale. = FALSE)
pca_data <- as.data.frame(pca_result$x[,1:2])
pca_data$Cluster <- covid_data$Cluster
pca_data$Risk <- covid_data$Risk
```

```r
# Plot PCA results with improvements
pca_plot <- ggplot(pca_data, aes(x = PC1, y = PC2, color = Risk)) +
  geom_jitter(alpha = 0.7, size = 2, width = 0.1, height = 0.1) +
  stat_ellipse(aes(fill = Risk), type = "norm", level = 0.95, geom = "polygon", alpha
= 0.2) +
  labs(title = "COVID-19 Symptom Clusters",
       x = "Principal Component 1",
       y = "Principal Component 2") +
  theme_minimal() +
  theme(
    plot.title = element_text(face = "bold", size = 16),
    axis.title = element_text(face = "bold", size = 12),
    legend.title = element_text(face = "bold"),
    panel.grid.major = element_line(color = "gray90"),
    panel.grid.minor = element_line(color = "gray95")
  ) +
  scale_color_brewer(palette = "Dark2") +
  scale_fill_brewer(palette = "Dark2") +
  coord_fixed(ratio = 1)

print(pca_plot)
```



```r
library(forcats)
```

```r
# Prepare the data
cluster_centers <- as.data.frame(kmeans_result$centers)
cluster_centers$Cluster <- as.factor(1:k)

cluster_centers_long <- cluster_centers %>%
  pivot_longer(cols = -Cluster, names_to = "Symptom", values_to = "Value") %>%
  mutate(Symptom = fct_reorder(Symptom, Value))  # Reorder symptoms by overall import
ance



# Calculate overall importance for each symptom
symptom_importance <- cluster_centers_long %>%
  group_by(Symptom) %>%
  summarize(OverallImportance = mean(abs(Value))) %>%
  arrange(desc(OverallImportance))



# Order symptoms by overall importance
cluster_centers_long$Symptom <- factor(cluster_centers_long$Symptom,
                                       levels = symptom_importance$Symptom)



# Create the enhanced plot
feature_importance_plot <- ggplot(cluster_centers_long, aes(x = Symptom, y = Value, f
ill = Cluster)) +
  geom_bar(stat = "identity", position = position_dodge(width = 0.9), width = 0.8) +
  geom_hline(yintercept = 0, linetype = "dashed", color = "gray50") +
  coord_flip() +  # Flip coordinates for horizontal bars
  facet_wrap(~ Cluster, nrow = 1) +  # Separate plot for each cluster
  labs(title = "Feature Importance by Cluster",
       subtitle = "Normalized values of symptoms for each cluster",
       x = "Symptoms",
       y = "Normalized Value",
       fill = "Risk Level") +
  theme_minimal(base_size = 12) +
  theme(
    plot.title = element_text(face = "bold", size = 16),
    plot.subtitle = element_text(size = 12, color = "gray30"),
    axis.title = element_text(face = "bold"),
    axis.text.y = element_text(size = 10),
    legend.position = "top",
    panel.grid.major.y = element_blank(),
    panel.grid.minor.y = element_blank(),
    strip.text = element_text(face = "bold")
  ) +
  scale_fill_brewer(palette = "Set2") +
  scale_y_continuous(limits = c(-1, 1), breaks = seq(-1, 1, 0.5))

print(feature_importance_plot)
```



Conclusion:

From this experiment, I successfully segmented patients into three distinct risk groups—Low, Moderate, and High—using K-means clustering on the COVID-19 symptoms dataset.

The optimal number of clusters (k=3) was determined through the Elbow Method, and key features such as fever, tiredness, dry cough, and breathing difficulty were selected for clustering.

The results highlighted clear patient segmentation based on symptom severity. The final PCA visualization demonstrated distinct clusters, offering valuable insights for healthcare applications, such as patient segmentation and risk assessment.