

Ανάλυση Δεδομένων με την χρήση του RShiny

Αλβέρτο Ντρέου

31/05/2024

Περίληψη

Στην παρούσα αναφορά περιγράφεται ο σχεδιασμός, η υλοποίηση και η ανάλυση δεδομένων για μια εφαρμογή που αναπτύχθηκε χρησιμοποιώντας τη γλώσσα προγραμματισμού R και χρησιμοποιεί 2D visualization για τα δεδομένα και 2 αλγορίθμους μηχανικής μάθησης που αφορούν την κατηγοροποίηση και την ομαδοποίηση των δεδομένων. Στο τέλος της εργασίας παρατίθενται τα συμπεράσματα και τα αποτελέσματα της εργασίας.

Περιεχόμενα

1	Εισαγωγή	2
1.1	Σκοπός	2
1.2	Δομή Αναφοράς	2
2	Σχεδιασμός της Εφαρμογής	3
2.1	Απαιτήσεις και Προδιαγραφές	3
2.2	Αρχιτεκτονική Σχεδίαση	3
3	Υλοποίηση	4
3.1	Περιγραφή Κώδικα	4
3.2	Βασικές Λειτουργίες	5
4	Αποτελέσματα των Αναλύσεων	6
4.1	Στατιστικά Δεδομένα	6
4.2	Γραφήματα	6
5	Συμπεράσματα	10
A'	Παραρτήματα	11
A'.1	Κώδικας σε R	11

Κεφάλαιο 1

Εισαγωγή

1.1 Σκοπός

Ο σκοπός της παρούσας εργασίας είναι η ανάπτυξη μιας διαδραστικής εφαρμογής ανάλυσης δεδομένων σε R χρησιμοποιώντας το πακέτο Shiny. Η εφαρμογή επιτρέπει την επεξεργασία, οπτικοποίηση και ανάλυση δεδομένων μέσω ενός φιλικού περιβάλλοντος χρήστη. Μετά την υλοποίηση της εφαρμογής πρέπει να αναπτύξουμε και να διανέμουμε την εργασία μέσω του δοσχερ. Στην τελική αναφορά της εργασίας πρέπει να συμπεριλάβουμε ένα UML διάγραμμα που θα απεικονίζει την αρχιτεκτονική της εφαρμογής και της διεπαφής χρήστη.

1.2 Δομή Αναφοράς

Η αναφορά χωρίζεται σε έξι κύρια κεφάλαια: Εισαγωγή, Σχεδιασμός της Εφαρμογής, Υλοποίηση, Αποτελέσματα των Αναλύσεων, Συμπεράσματα.

Κεφάλαιο 2

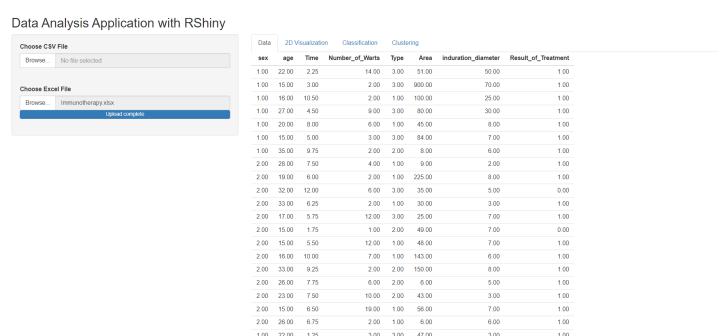
Σχεδιασμός της Εφαρμογής

2.1 Απαιτήσεις και Προδιαγραφές

Η εφαρμογή πρέπει να είναι σε θέση να διαβάζει δεδομένα από αρχεία CSV και Excel, να κάνει οπτικοποίηση των δεδομένων, να εκτελεί 2D visualization με τις μεθόδους μείωσης διάστασης και απεικόνιση με PCA και t-sne, να εφαρμόζει αλγόριθμους κατηγοριοποίησης με τις μεθόδους KNN και Logistic regression και ομαδοποίησης με k-means και Hierachical.

2.2 Αρχιτεκτονική Σχεδίαση

Η αρχιτεκτονική της εφαρμογής περιλαμβάνει τρία κύρια μέρη: την ανάγνωση δεδομένων, την επεξεργασία και την οπτικοποίηση.



The screenshot shows a web application interface titled "Data Analysis Application with RShiny". On the left, there are two file upload sections: "Choose CSV File" and "Choose Excel File", both with "Browse" buttons and "Upload" links. The main area displays a data table with the following columns: sex, age, Time, Number_of_Warts, Type, Area, Induration_diameter, and Result_of_Treatment. The table contains 20 rows of data.

sex	age	Time	Number_of_Warts	Type	Area	Induration_diameter	Result_of_Treatment
1.00	22.00	2.25	14.00	3.00	51.00	50.00	1.00
1.00	15.00	3.00	2.00	3.00	600.00	70.00	1.00
1.00	16.00	10.50	2.00	1.00	100.00	25.00	1.00
1.00	27.00	4.50	9.00	3.00	30.00	30.00	1.00
1.00	20.00	6.00	6.00	1.00	45.00	8.00	1.00
1.00	15.00	5.00	3.00	3.00	54.00	7.00	1.00
1.00	35.00	9.75	2.00	2.00	8.00	6.00	1.00
2.00	26.00	7.50	4.00	1.00	9.00	2.00	1.00
2.00	19.00	6.00	2.00	1.00	225.00	8.00	1.00
2.00	32.00	12.00	6.00	3.00	35.00	5.00	0.00
2.00	30.00	6.25	2.00	1.00	30.00	5.00	1.00
2.00	17.00	5.75	12.00	3.00	25.00	7.00	1.00
2.00	15.00	1.75	1.00	2.00	40.00	7.00	0.00
2.00	15.00	5.50	12.00	1.00	48.00	7.00	1.00
2.00	16.00	10.00	7.00	1.00	143.00	6.00	1.00
2.00	31.00	9.25	2.00	2.00	150.00	8.00	1.00
2.00	26.00	1.75	6.00	2.00	6.00	5.00	1.00
2.00	23.00	7.50	10.00	2.00	43.00	3.00	1.00
2.00	15.00	6.50	19.00	1.00	56.00	7.00	1.00
2.00	26.00	6.75	2.00	1.00	6.00	6.00	1.00
1.00	27.00	1.25	3.00	3.00	47.00	3.00	1.00

Σχήμα 2.1: Διάγραμμα Αρχιτεκτονικής Εφαρμογής

Κεφάλαιο 3

Υλοποίηση

3.1 Περιγραφή Κώδικα

Ο κώδικας υλοποιήθηκε σε R και χρησιμοποιεί ως κύριο πακέτο του την RShiny για τη δημιουργία διαδραστικού περιβάλλοντος χρήστη. Ο κώδικας περιλαμβάνει τις ακόλουθες ενότητες: ανάγνωση δεδομένων, επεξεργασία και ανάλυση, οπτικοποίηση. Παρατίθεται κάτω ο αναλυτικός κώδικας:

```
library(shiny) library(readr) library(readxl) library(ggplot2) library(plotly)
library(Rtsne) library(caret) library(cluster) library(factoextra)
ui <- fluidPage( titlePanel("Data Analysis Application with RShiny"),
  sidebarLayout( sidebarPanel( fileInput("file1", "Choose CSV File", ac-
    cept = ".csv"), fileInput("file2", "Choose Excel File", accept = ".xlsx"),
  ),
  mainPanel( tabsetPanel( tabPanel("Data", tableOutput("contents")),
    tabPanel("2D Visualization", selectInput("method", "Choose Method",
    choices = c("PCA", "t-SNE")), actionButton("plot", "Plot"), plotlyOut-
    put("plot2d" ) ),
    tabPanel("Classification", selectInput("classifier", "Choose Classifier", choices
    = c("k-NN", "Logistic Regression")), numericInput("k", "k for k-NN", 3,
    min = 1, max = 20), actionButton("classify", "Classify"), verbatimTextOutput("classification_result"),
    tabPanel("Clustering", selectInput("clustering", "Choose Clustering Method",
    choices = c("k-means", "Hierarchical")), numericInput("k_clusters", "Number of clusters(k)", 3, min
    = 1, max = 10), actionButton("cluster", "Cluster"), plotOutput("clustering_result"))
  ) ) )
server <- function(input, output) data <- reactive( req(input$file1) if(!is.null(input$file1))
  read_csv(input$file1$datapath) else if(!is.null(input$file2)) read_excel(input$file2$datapath) )
output$contents <- renderTable(data())
output$plot2d <- renderPlotly(req(input$plot) df <- data() labels <- df[,
```

```

ncol(df)] features <- df[, -ncol(df)]
  if (inputmethod == "PCA") pca_result <- prcomp(features, center = TRUE, scale. = TRUE,
1:2], Label = labels) colnames(plot_data) <- c("PC1", "PC2", "Label") p <-
-ggplot(plot_data, aes(x = PC1, y = PC2, color = Label)) + geom_point() else if (inputmethod
== "t-SNE") tsne_result <- Rtsne(features, dims = 2) plot_data <- data.frame(tsne_result$Y,
Label = labels) colnames(plot_data) <- c("Dim1", "Dim2", "Label") p <-
-ggplot(plot_data, aes(x = Dim1, y = Dim2, color = Label)) + geom_point()
  ggplotly(p) )
  outputclassification_result <- renderPrint(req(inputclassify) df <- data()
labels <- df[, ncol(df)] features <- df[, -ncol(df)]
  trainIndex <- createDataPartition(labels, p = 0.8, list = FALSE) train-
Data <- features[trainIndex,] testData <- features[-trainIndex,] trainLabels
<- labels[trainIndex] testLabels <- labels[-trainIndex]
  if (inputclassifier == "k-NN") model <- train(trainData, trainLabels, method = "knn",
else if (inputclassifier == "LogisticRegression") model <- train(trainData, trainLabels, meth
  predictions <- predict(model, testData) confusionMatrix(predictions, test-
Labels) )
  outputclustering_result <- renderPlot(req(inputcluster) df <- data()
features <- df[, -ncol(df)]
  if (inputclustering == "k-means") km_result <- kmeans(features, centers = inputkcluster
features) else if (inputclustering == "Hierarchical") hclust_result <- hclust(dist(features)) plot
  shinyApp(ui, server)

```

3.2 Βασικές Λειτουργίες

Η εφαρμογή εκτελεί λειτουργίες όπως:

- Ανάγνωση δεδομένων από CSV και Excel
- 2D οπτικοποιήσεις βασισμένες σε 2 αλγόριθμους μείωσης διάστασης και απεικόνιση με PCA και t-SNE
- Κατηγοριοποίηση δεδομένων με k-NN και Logistic Regression
- Ομαδοποίηση δεδομένων με k-means και Hierarchical

Κεφάλαιο 4

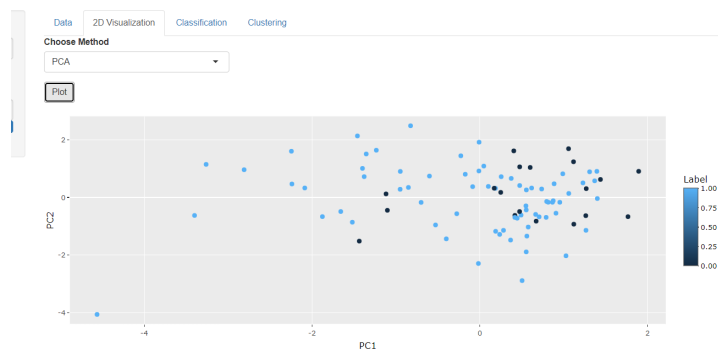
Αποτελέσματα των Αναλύσεων

4.1 Στατιστικά Δεδομένα

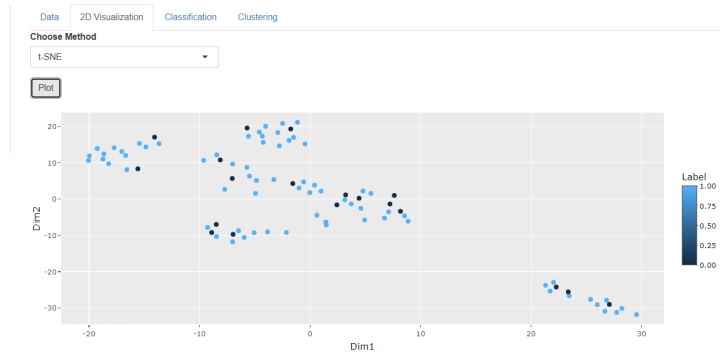
Τα δεδομένα που χρησιμοποιήθηκαν αφορούν την immunotherapy και αποτελούνται από 8 κατηγορίες που μια από αυτές είναι το αποτέλεσμα. Οι 7 κατηγορίες αφορούν το φύλο, την ηλικία, τον χρόνο, το number of warts, τον τύπο, την περιοχή και την διάμετρο. Η τελευταία που είναι και το αποτέλεσμα που δηλώνει αν ο ασθενής πάσχει από την ασθένεια ή όχι είναι το result of treatment.

4.2 Γραφήματα

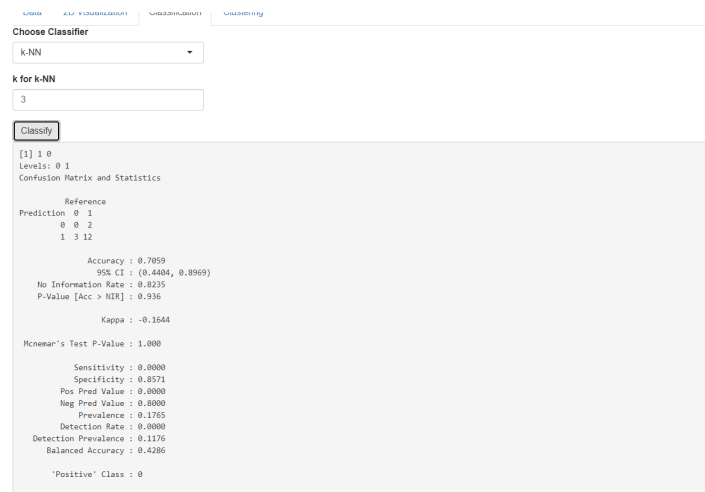
Παρουσιάζονται τα γραφήματα που δημιουργήθηκαν από τα δεδομένα:



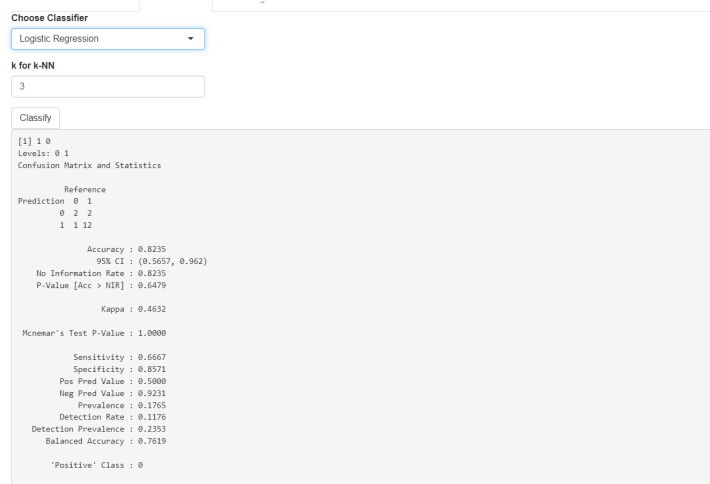
Σχήμα 4.1: 2D visualization PCA



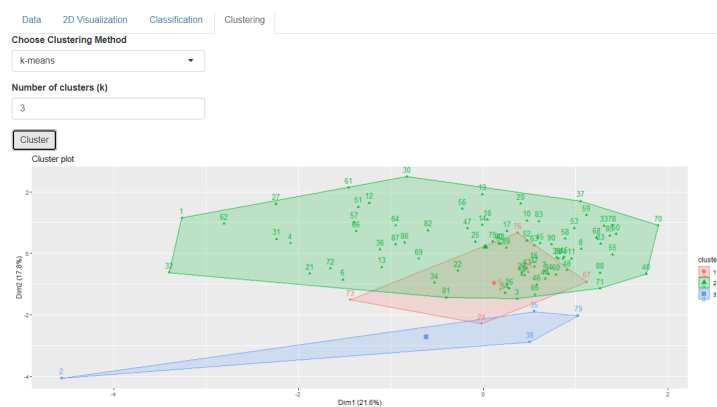
Σχήμα 4.2: 2D visualization T-sne



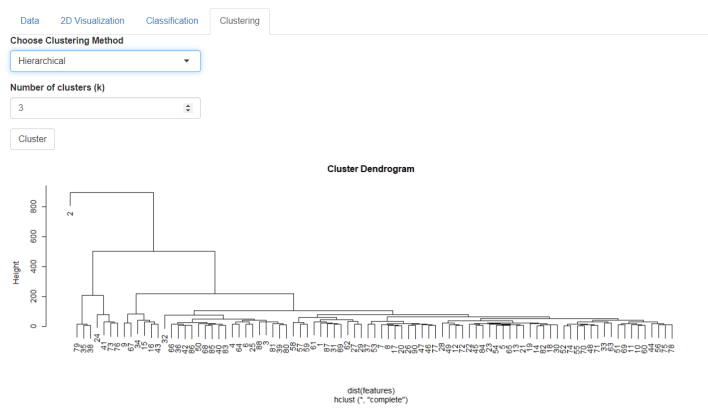
Σχήμα 4.3: Classification KNN



Σχήμα 4.4: Classification Logistic Regression



Σχήμα 4.5: Clustering K-means



Σχήμα 4.6: Clustering Hierarchial

Κεφάλαιο 5

Συμπεράσματα

Η ανάλυση των δεδομένων ανέδειξε σημαντικά προβλήματα απόδοσης, κυρίως λόγω της ανισοβάρειας των κατηγοριών. Αν και τα μοντέλα που χρησιμοποιήσαμε έχουν αρκετή ακρίβεια το θέμα σε αυτά τα δεδομένα είναι ότι η μία κλάση υπερσχυρεί της αλλής οπότε η απόδοση των αλγορίθμων διαφέρει π.χ. στο Sensitivity του KNN είναι 0 σε σχέση με του Logistic Regression που είναι 0.6667 και η ακρίβεια είναι υψολότερη σε σχέση με το KNN.

Παράρτημα Α΄

Παραρτήματα

Α΄.1 Κώδικας σε R

```
library(shiny) library(readr) library(readxl) library(ggplot2) library(plotly)
library(Rtsne) library(caret) library(cluster) library(factoextra)
  ui <- fluidPage( titlePanel("Data Analysis Application with RShiny"),
    sidebarLayout( sidebarPanel( fileInput("file1", "Choose CSV File", ac-
      cept = ".csv"), fileInput("file2", "Choose Excel File", accept = ".xlsx"),
    ),
    mainPanel( tabsetPanel( tabPanel("Data", tableOutput("contents")),
      tabPanel("2D Visualization", selectInput("method", "Choose Method",
        choices = c("PCA", "t-SNE")), actionButton("plot", "Plot"), plotlyOut-
        put("plot2d" ) ),
      tabPanel("Classification", selectInput("classifier", "Choose Classifier", choices
        = c("k-NN", "Logistic Regression")), numericInput("k", "k for k-NN", 3,
        min = 1, max = 20), actionButton("classify", "Classify"), verbatimTextOutput("classification_result"),
      tabPanel("Clustering", selectInput("clustering", "Choose Clustering Method",
        choices = c("k-means", "Hierarchical")), numericInput("clusters", "Number of clusters(k)", 3, min
        = 1, max = 10), actionButton("cluster", "Cluster"), plotOutput("clustering_result"))
    ) ) )
  server <- function(input, output) data <- reactive( req(input$file1) if(!is.null(input$file1))
    read_csv(input$file1$datapath) else if(!is.null(input$file2)) read_excel(input$file2$datapath) )
  output$contents <- renderTable(data())
  output$plot2d <- renderPlotly(req(input$plot) df <- data() labels <- df[,
    ncol(df)] features <- df[, -ncol(df)]
    if (input$method == "PCA") pca_result <- prcomp(features, center = TRUE, scale. = TRUE)
    else if (input$method == "t-SNE") tsne_result <- t_sne(features)
    else if (input$method == "k-NN") knn_result <- knn(train = features, test = features, k = input$k)
    else if (input$method == "Logistic Regression") logit_result <- glm(features, labels, family = "binomial")
    else if (input$method == "k-means") kmeans_result <- kmeans(features, input$clusters)
    else if (input$method == "Hierarchical") hclust_result <- hclust(features)
    else if (input$method == "Classification") {
      if (input$classifier == "k-NN") knn_result <- knn(train = features, test = features, k = input$k)
      else if (input$classifier == "Logistic Regression") logit_result <- glm(features, labels, family = "binomial")
    }
    if (input$method == "PCA") {
      plot_data <- as.data.frame(pca_result$x[, 1:2], Label = labels)
      colnames(plot_data) <- c("PC1", "PC2", "Label")
      p <- ggplot(plot_data, aes(x = PC1, y = PC2, color = Label)) + geom_point()
    }
    else if (input$method == "t-SNE") {
      plot_data <- as.data.frame(tsne_result[, 1:2], Label = labels)
      colnames(plot_data) <- c("t-SNE1", "t-SNE2", "Label")
      p <- ggplot(plot_data, aes(x = t-SNE1, y = t-SNE2, color = Label)) + geom_point()
    }
    else if (input$method == "k-NN") {
      plot_data <- as.data.frame(knn_result, Label = labels)
      colnames(plot_data) <- c("k-NN", "Label")
      p <- ggplot(plot_data, aes(x = k-NN, y = Label)) + geom_point()
    }
    else if (input$method == "Logistic Regression") {
      plot_data <- as.data.frame(logit_result, Label = labels)
      colnames(plot_data) <- c("Logistic Regression", "Label")
      p <- ggplot(plot_data, aes(x = Logistic Regression, y = Label)) + geom_point()
    }
    else if (input$method == "k-means") {
      plot_data <- as.data.frame(kmeans_result, Label = labels)
      colnames(plot_data) <- c("k-means", "Label")
      p <- ggplot(plot_data, aes(x = k-means, y = Label)) + geom_point()
    }
    else if (input$method == "Hierarchical") {
      plot_data <- as.data.frame(hclust_result, Label = labels)
      colnames(plot_data) <- c("Hierarchical", "Label")
      p <- ggplot(plot_data, aes(x = Hierarchical, y = Label)) + geom_point()
    }
    else if (input$method == "Classification") {
      if (input$classifier == "k-NN") {
        plot_data <- as.data.frame(knn_result, Label = labels)
        colnames(plot_data) <- c("k-NN", "Label")
        p <- ggplot(plot_data, aes(x = k-NN, y = Label)) + geom_point()
      }
      else if (input$classifier == "Logistic Regression") {
        plot_data <- as.data.frame(logit_result, Label = labels)
        colnames(plot_data) <- c("Logistic Regression", "Label")
        p <- ggplot(plot_data, aes(x = Logistic Regression, y = Label)) + geom_point()
      }
    }
    p
  )
```

```

== "t-SNE") tsne_result <- Rtsne(features, dims = 2) plot_data <- data.frame(tsne_result$Y,
Label = labels) colnames(plot_data) <- c("Dim1", "Dim2", "Label") p <-
-ggplot(plot_data, aes(x = Dim1, y = Dim2, color = Label)) + geom_point()
ggplotly(p) )
output_classification_result <- renderPrint(req(inputclassify) df <- data()
labels <- df[, ncol(df)] features <- df[, -ncol(df)]
trainIndex <- createDataPartition(labels, p = 0.8, list = FALSE) train-
Data <- features[trainIndex,] testData <- features[-trainIndex,] trainLabels
<- labels[trainIndex] testLabels <- labels[-trainIndex]
if (input_classifier == "k-NN") model <- train(trainData, trainLabels, method = "knn",
else if (input_classifier == "LogisticRegression") model <- train(trainData, trainLabels, meth
predictions <- predict(model, testData) confusionMatrix(predictions, test-
Labels) )
output_clustering_result <- renderPlot(req(inputcluster) df <- data()
features <- df[, -ncol(df)]
if (input_clustering == "k-means") km_result <- kmeans(features, centers = input_k_cluster
features) else if (input_clustering == "Hierarchical") hclust_result <- hclust(dist(features)) plot
shinyApp(ui, server)

```