Data Analysis Application Report

Maria Balasopoulou AM inf2021150, Theodora Matsaridou AM inf2021136

May 29, 2024

1 Link of the application

http://192.168.1.15:3838

2 Introduction

The Data Analysis Application presented in this report serves as a tool for users to analyze data from CSV or Excel files. Its main functionalities include data loading, variable selection, data visualization, classification using Decision Tree or Logistic Regression, and clustering using K-Means or Hierarchical Clustering.

3 Design Overview

The application is built using RShiny, with a user interface (UI) and server components. The UI consists of multiple tabs, each dedicated to a specific functionality. Users can load their data file, select variables, visualize data, perform classification, and conduct clustering analysis. The server components handle the logic behind each functionality and interact with the UI elements.

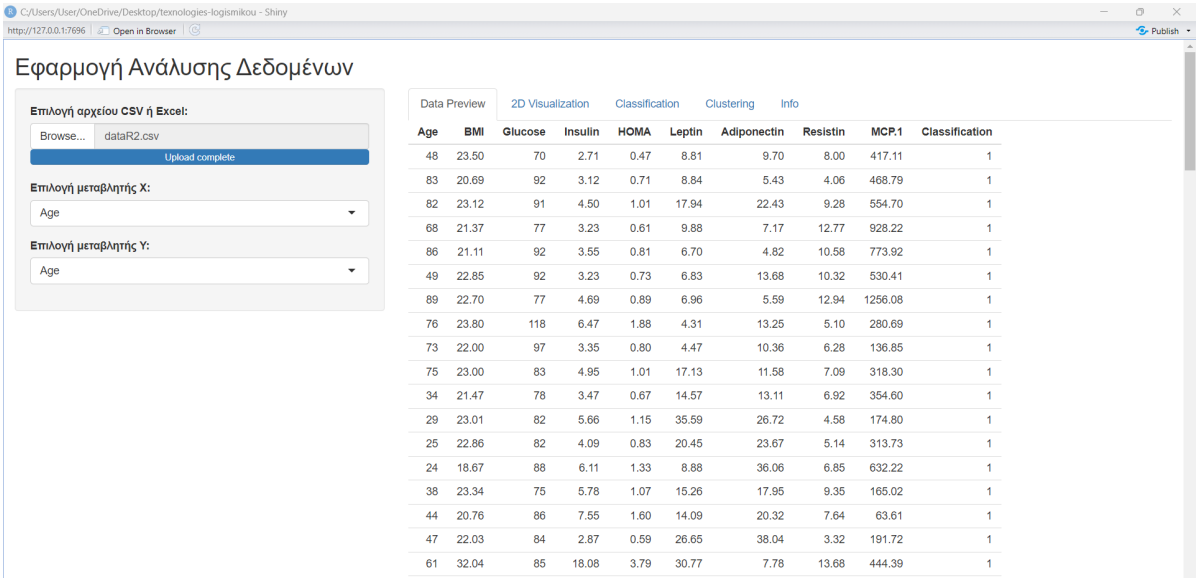
4 Implementation Details

The implementation of the application utilizes various R packages, including `shiny`, `rpart`, `ggplot2`, `dplyr`, and `cluster`. Key implementation details for each module are as follows:

4.1 Data Loading

The data loading module allows users to upload CSV or Excel files. Upon file selection, the application reads the data using `read.csv` or `read_excel` functions from the `readr` package.

```
inFile <- input$file
if (is.null(inFile))
```



Age	BMI	Glucose	Insulin	HOMA	Leptin	Adiponectin	Resistin	MCP.1	Classification
48	23.50	70	2.71	0.47	8.81	9.70	8.00	417.11	1
83	20.69	92	3.12	0.71	8.84	5.43	4.06	468.79	1
82	23.12	91	4.50	1.01	17.94	22.43	9.28	554.70	1
68	21.37	77	3.23	0.61	9.88	7.17	12.77	928.22	1
86	21.11	92	3.55	0.81	6.70	4.82	10.58	773.92	1
49	22.85	92	3.23	0.73	6.83	13.68	10.32	530.41	1
89	22.70	77	4.69	0.89	6.96	5.59	12.94	1256.08	1
76	23.80	118	6.47	1.88	4.31	13.25	5.10	280.69	1
73	22.00	97	3.35	0.80	4.47	10.36	6.28	136.85	1
75	23.00	83	4.95	1.01	17.13	11.58	7.09	318.30	1
34	21.47	78	3.47	0.67	14.57	13.11	6.92	354.60	1
29	23.01	82	5.66	1.15	35.59	26.72	4.58	174.80	1
25	22.86	82	4.09	0.83	20.45	23.67	5.14	313.73	1
24	18.67	88	6.11	1.33	8.88	36.06	6.85	632.22	1
38	23.34	75	5.78	1.07	15.26	17.95	9.35	165.02	1
44	20.76	86	7.55	1.60	14.09	20.32	7.64	63.61	1
47	22.03	84	2.87	0.59	26.65	38.04	3.32	191.72	1
61	32.04	85	18.08	3.79	30.77	7.78	13.68	444.39	1

Figure 1: load file example

```
  return(NULL)
df <- read.csv(inFile$datapath)
```

4.2 Variable Selection

Users can select the X and Y variables for data visualization, classification, and clustering. The application dynamically generates dropdown menus based on the columns of the loaded dataset.

```
# X variable selection
  output$x_var_selector <- renderUI({
    req(data())
    selectInput("x_var", "Choose X variable:", choices = colnames(data()))
  })
# Y variable selection
  output$y_var_selector <- renderUI({
    req(data())
    selectInput("y_var", "Choose Y variable:", choices = colnames(data()))
  })
```

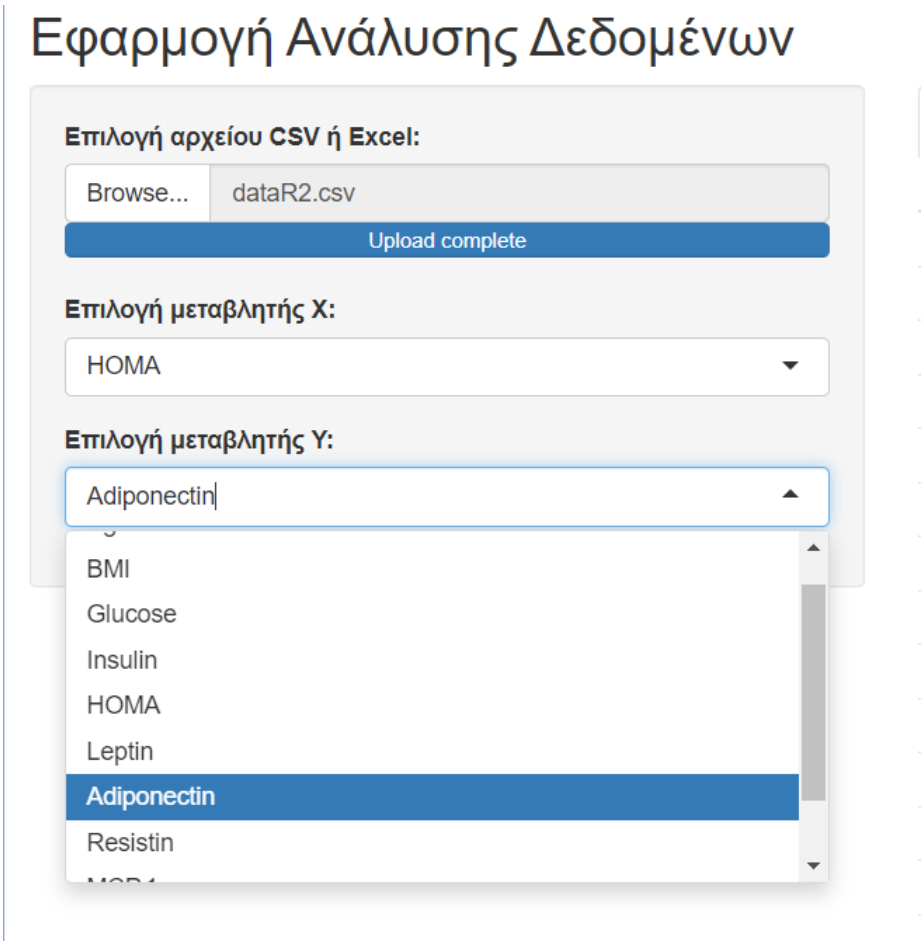


Figure 2: variables selection example

4.3 Data Visualization

The data visualization module generates a scatter plot using `ggplot2` to visualize the relationship between the selected variables.

```
ggplot(data(), aes(x = !!sym(x_var), y = !!sym(y_var))) +
  geom_point() +
  labs(x = x_var, y = y_var) +
  theme_minimal()
```

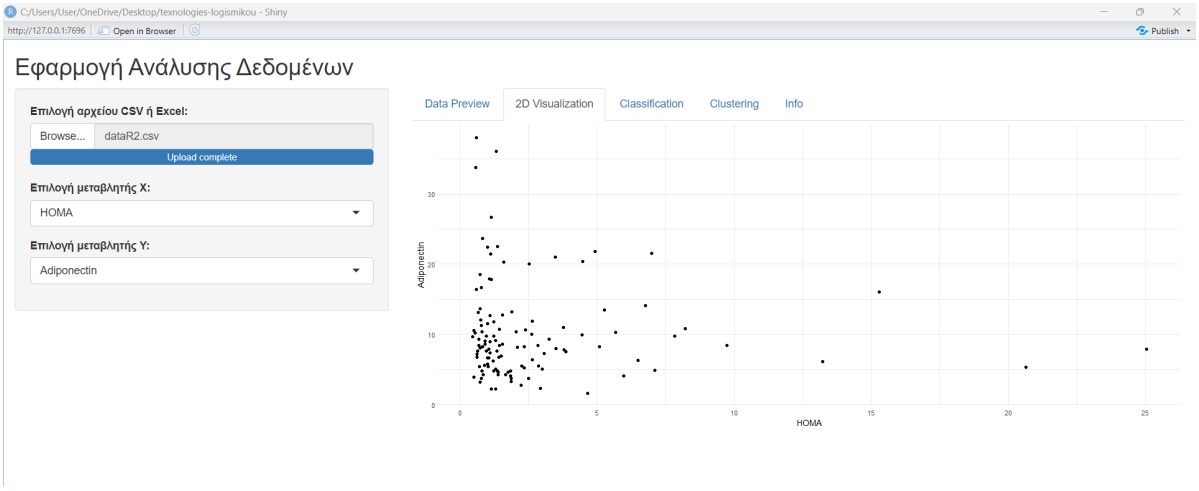


Figure 3: 2D Visualization example

4.4 Classification

Users can choose between Decision Tree and Logistic Regression algorithms for classification. The models are trained using `rpart` and `glm` functions, respectively.

```
# Classification with Decision Tree or Logistic Regression
  accuracy <- reactiveValues(dt = NULL, lr = NULL)

  observeEvent(input$classify_button, {
    req(input$file)
    req(input$x_var)
    req(input$y_var)
    req(input$classification_algorithm)

    if (input$classification_algorithm == "Decision Tree") {
      data_processed <- data() %>%
        mutate(y_category = cut(.data[[input$y_var]], breaks = 3, labels = c("Low", "Medium"
      dt_model <- rpart(y_category ~ ., data = data_processed)
      prediction <- predict(dt_model, newdata = data_processed, type = "class")
      accuracy$dt <- sum(prediction == data_processed$y_category) / nrow(data_processed)
      output$classification_result <- renderPrint({
        cat("Decision Tree Model Summary:\n")
        print(summary(dt_model))
        cat("\nAccuracy:", accuracy$dt, "\n")
      })
    } else if (input$classification_algorithm == "Logistic Regression") {
      threshold <- 0.5
```

```
    df <- data() %>%
      mutate(y_category = cut(.data[[input$y_var]], breaks = 3, labels = c("Low", "Medium"
    lr_model <- glm(formula = as.formula(paste("y_category ~ .")), data = df, family = bin
    prediction <- predict(lr_model, newdata = df, type = "response")
    prediction <- ifelse(prediction > threshold, "High", ifelse(prediction < threshold, "L
    accuracy$lr <- sum(prediction == df$y_category) / nrow(df)
    output$classification_result <- renderPrint({
      cat("Logistic Regression Model Summary:\n")
      print(summary(lr_model))
      cat("\nAccuracy:", accuracy$lr, "\n")
    })
  }
})

# Comparison of algorithm accuracies
output$accuracy_comparison <- renderPrint({
  req(input$classify_button)

  if (!is.null(accuracy$dt) && !is.null(accuracy$lr)) {
    if (accuracy$dt > accuracy$lr) {
      cat("Decision Tree has higher accuracy.\n")
    } else if (accuracy$dt < accuracy$lr) {
      cat("Logistic Regression has higher accuracy.\n")
    } else {
      cat("Both algorithms have the same accuracy.\n")
    }
  }
})
```

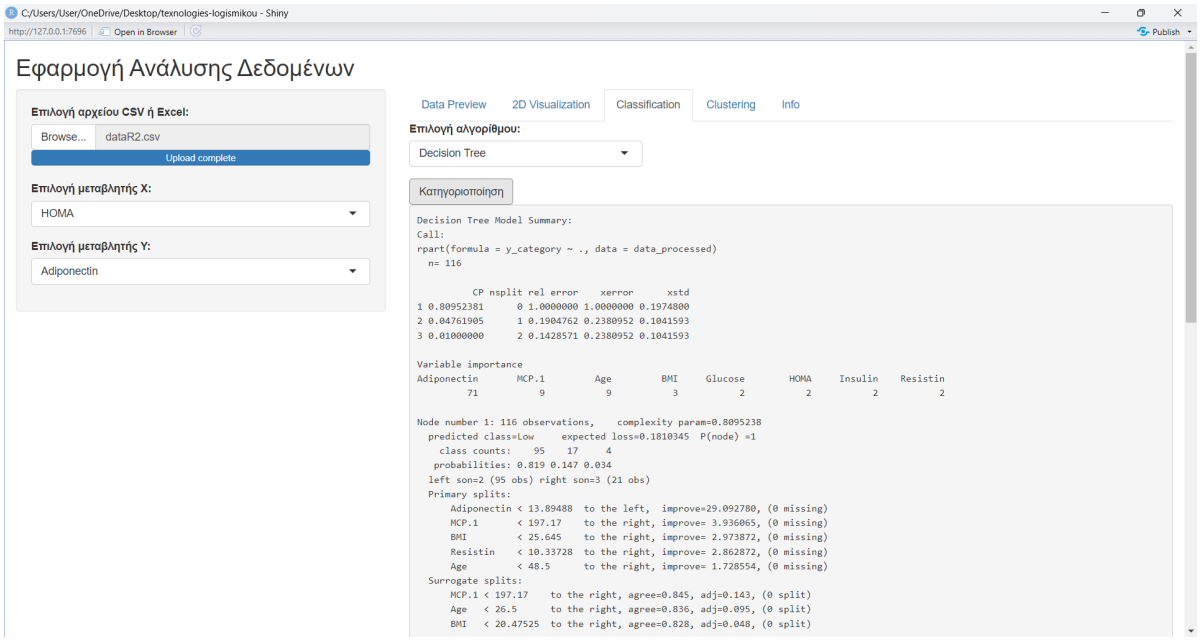


Figure 4: decision tree example 1

```
      BMI   < 20.47525  to the right, agree=0.828, adj=0.048, (0 split)

Node number 2: 95 observations
  predicted class=Low     expected loss=0  P(node) =0.8189655
    class counts:    95     0     0
   probabilities: 1.000 0.000 0.000

Node number 3: 21 observations,    complexity param=0.04761905
  predicted class=Medium  expected loss=0.1904762  P(node) =0.1810345
    class counts:     0    17     4
   probabilities: 0.000 0.810 0.190
  left son=6 (14 obs) right son=7 (7 obs)
  Primary splits:
      Adiponectin < 22.12789  to the left,  improve=3.047619, (0 missing)
      Glucose     < 89.5      to the right, improve=2.031746, (0 missing)
      Resistin    < 7.085     to the right, improve=2.031746, (0 missing)
      Leptin      < 23.37     to the left,  improve=1.190476, (0 missing)
      HOMA        < 1.350499  to the right, improve=1.142857, (0 missing)
  Surrogate splits:
      Age      < 31.5      to the right, agree=0.810, adj=0.429, (0 split)
      Glucose  < 85        to the right, agree=0.762, adj=0.286, (0 split)
      Insulin  < 3.0285    to the right, agree=0.762, adj=0.286, (0 split)
      HOMA     < 0.5977537 to the right, agree=0.762, adj=0.286, (0 split)
      Resistin < 3.835     to the right, agree=0.762, adj=0.286, (0 split)

Node number 6: 14 observations
  predicted class=Medium  expected loss=0  P(node) =0.1206897
    class counts:     0    14     0
   probabilities: 0.000 1.000 0.000

Node number 7: 7 observations
  predicted class=High    expected loss=0.4285714  P(node) =0.06034483
    class counts:     0     3     4
   probabilities: 0.000 0.429 0.571

n= 116

node), split, n, loss, yval, (yprob)
      * denotes terminal node
```

Figure 5: decision tree example 2

```
Node number 3: 21 observations,    complexity param=0.04761905
  predicted class=Medium  expected loss=0.1904762  P(node) =0.1810345
    class counts:     0    17     4
   probabilities: 0.000 0.810 0.190
  left son=6 (14 obs) right son=7 (7 obs)
  Primary splits:
      Adiponectin < 22.12789  to the left,  improve=3.047619, (0 missing)
      Glucose     < 89.5      to the right, improve=2.031746, (0 missing)
      Resistin    < 7.085     to the right, improve=2.031746, (0 missing)
      Leptin      < 23.37     to the left,  improve=1.190476, (0 missing)
      HOMA        < 1.350499  to the right, improve=1.142857, (0 missing)
  Surrogate splits:
      Age      < 31.5      to the right, agree=0.810, adj=0.429, (0 split)
      Glucose  < 85        to the right, agree=0.762, adj=0.286, (0 split)
      Insulin  < 3.0285    to the right, agree=0.762, adj=0.286, (0 split)
      HOMA     < 0.5977537 to the right, agree=0.762, adj=0.286, (0 split)
      Resistin < 3.835     to the right, agree=0.762, adj=0.286, (0 split)

Node number 6: 14 observations
  predicted class=Medium  expected loss=0  P(node) =0.1206897
    class counts:     0    14     0
   probabilities: 0.000 1.000 0.000

Node number 7: 7 observations
  predicted class=High    expected loss=0.4285714  P(node) =0.06034483
    class counts:     0     3     4
   probabilities: 0.000 0.429 0.571

n= 116

node), split, n, loss, yval, (yprob)
      * denotes terminal node

1) root 116 21 Low (0.81896552 0.14655172 0.03448276)
  2) Adiponectin< 13.89487 95  0 Low (1.00000000 0.00000000 0.00000000) *
  3) Adiponectin>=13.89487 21  4 Medium (0.00000000 0.80952381 0.19047619)
    6) Adiponectin< 22.12789 14  0 Medium (0.00000000 1.00000000 0.00000000) *
    7) Adiponectin>=22.12789 7  3 High (0.00000000 0.42857143 0.57142857) *

Accuracy: 0.9741379
```

Figure 6: decision tree example 3

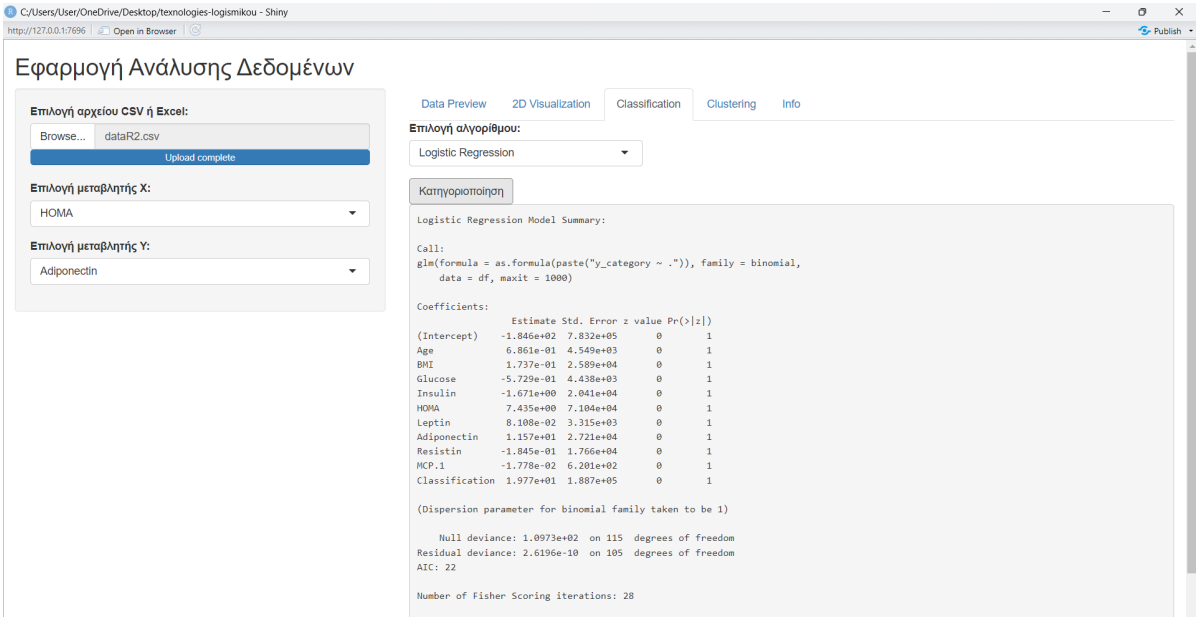


Figure 7: logistic regression example 1

```
    Null deviance: 1.0973e+02  on 115  degrees of freedom
Residual deviance: 2.6196e-10  on 105  degrees of freedom
AIC: 22

Number of Fisher Scoring iterations: 28


Accuracy: 0.8534483
```

```
Decision Tree έχει υψηλότερη ακρίβεια.
```

Figure 8: logistic regression example 2

4.5 Clustering

K-Means and Hierarchical Clustering algorithms are available for clustering analysis. They are implemented using `kmeans` and `hclust` functions, respectively.

```
# Clustering with K-Means or Hierarchical Clustering
  silhouette_scores <- reactiveValues(kmeans = NULL, hclust = NULL)

  observeEvent(input$run_clustering, {
    req(input$file)
    req(input$x_var)
    req(input$y_var)
    req(input$clustering_algorithm)

    if (input$clustering_algorithm == "K-Means") {
      kmeans_result <- kmeans(data()[, c(input$x_var, input$y_var)], centers = 3)
      cluster_labels <- kmeans_result$cluster
      silhouette_score <- silhouette(cluster_labels, dist(data()[, c(input$x_var, input$y_va
      silhouette_scores$kmeans <- mean(silhouette_score[, "sil_width"])
      output$clustering_result <- renderPrint({
        cat("K-Means Clustering Summary:\n")
        print(kmeans_result)
        cat("\nSilhouette Score:", silhouette_scores$kmeans, "\n")
      })
    } else if (input$clustering_algorithm == "Hierarchical Clustering") {
      hclust_result <- hclust(dist(data()[, c(input$x_var, input$y_var)]))
      cluster_labels <- cutree(hclust_result, k = 3)
      silhouette_score <- silhouette(cluster_labels, dist(data()[, c(input$x_var, input$y_va
      silhouette_scores$hclust <- mean(silhouette_score[, "sil_width"])
      output$clustering_result <- renderPrint({
        cat("Hierarchical Clustering Summary:\n")
        print(hclust_result)
        cat("\nSilhouette Score:", silhouette_scores$hclust, "\n")
      })
    }

    # Comparison of algorithm silhouette scores
    if (!is.null(silhouette_scores$kmeans) && !is.null(silhouette_scores$hclust)) {
      output$silhouette_comparison <- renderPrint({
        if (silhouette_scores$kmeans > silhouette_scores$hclust) {
          cat("K-Means has higher Silhouette Score.\n")
        } else if (silhouette_scores$kmeans < silhouette_scores$hclust) {
          cat("Hierarchical Clustering has higher Silhouette Score.\n")
        } else {
          cat("Both algorithms have the same Silhouette Score.\n")
        }
```

```
    })
  }
})
```

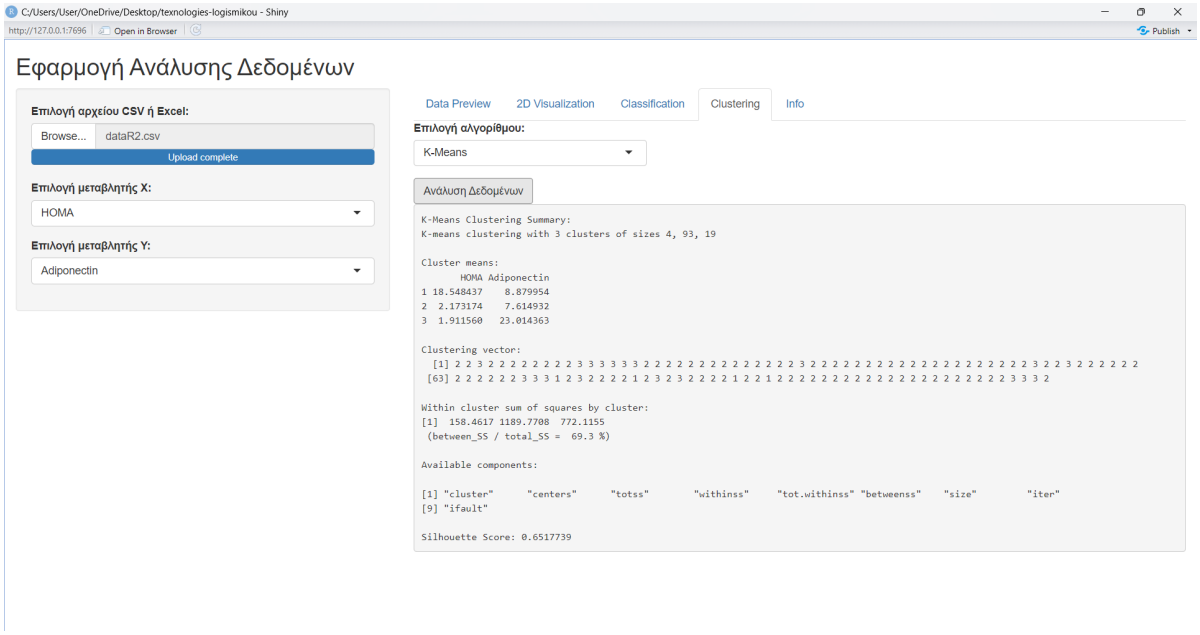


Figure 9: K means example

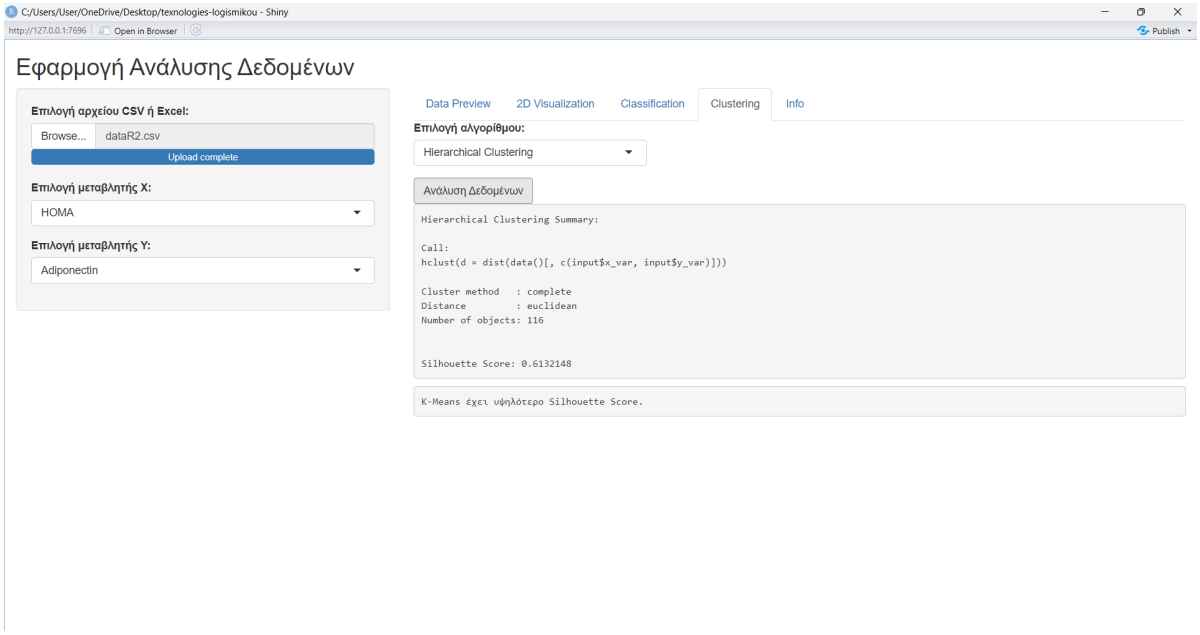


Figure 10: hierarchical clustering example

4.6 Info

the info tab is dedicated to the presentation information about the application, how it works and the development team.

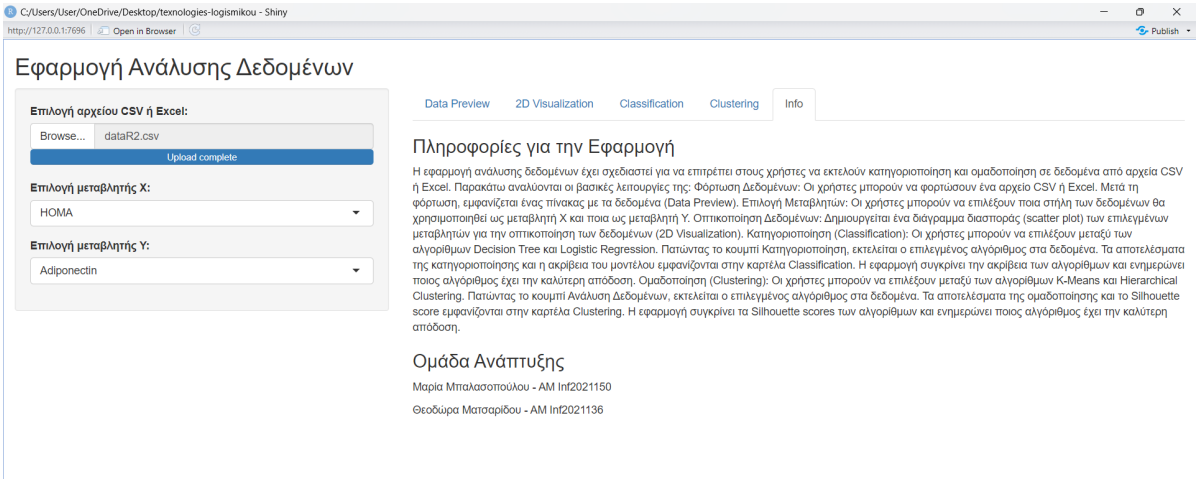


Figure 11: info example

5 Results of Analyses

The application provides results for classification accuracies and silhouette scores for clustering. These results are displayed to the user after performing the respective analyses.

6 Conclusions

The Data Analysis Application offers users a versatile platform for exploring and analyzing their data. Through its intuitive interface and powerful analytical capabilities, users can gain valuable insights and make data-driven decisions. Future enhancements could include additional algorithms for classification and clustering, as well as improved visualization options.

7 Contributions of Team Members

Maria focused on the implementation of data loading, variable selection, and clustering functionalities, while Theodora primarily worked on data visualization, classification, and result presentation. Both team members contributed to the design, testing, and documentation of the application.

8 UML Diagram

the UML diagram depicting the application and interface architecture user interface

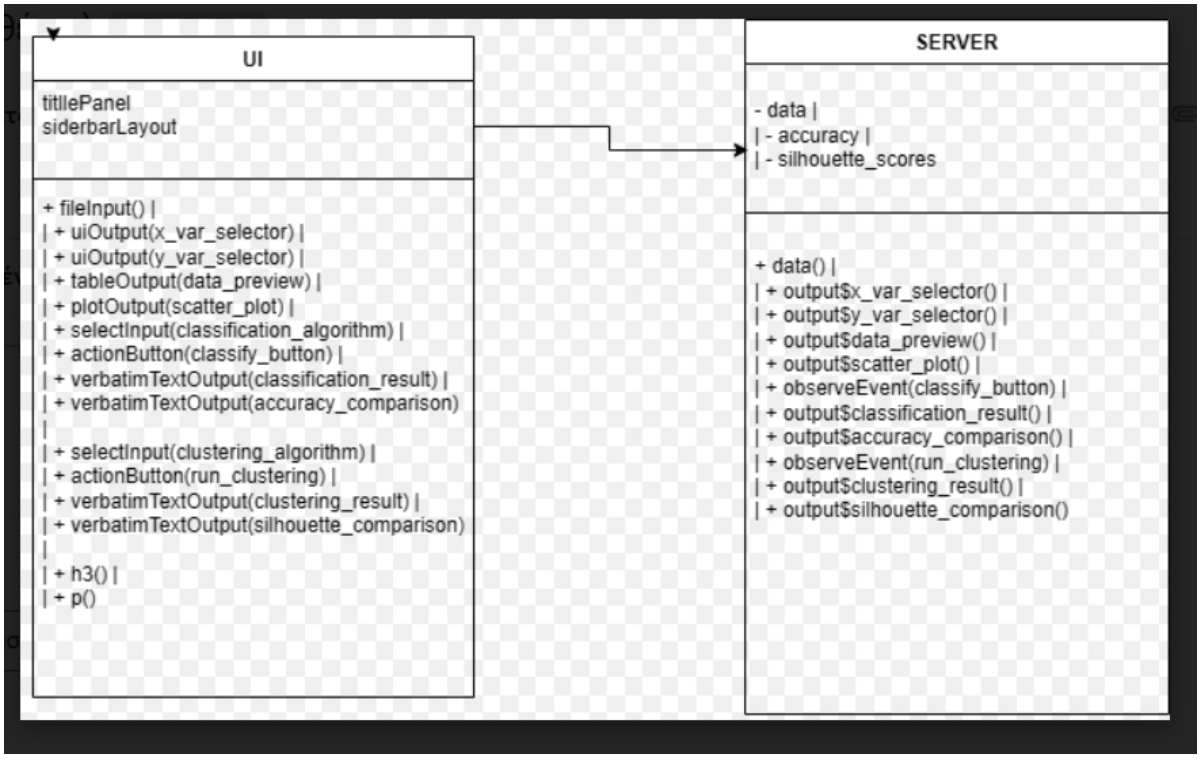


Figure 12: UML diagram

9 Software Release Lifecycle Waterfall

Software Release Lifecycle with the Waterfall Model for Data Analysis Application

1. Requirements Analysis: Using CSV and Excel files as a data source.Selecting variables for analysis.Creating scatter plots.Performing classification using Decision Tree and Logistic Regression.Performing clustering using K-Means and Hierarchical Clustering.Comparative analysis of results and algorithms. Requirements Documentation:We document all requirements and functions in a detailed requirements document, which includes:Data loading and preview functions.Variable selection capabilities for analysis.Categorization and clustering functions with corresponding results and comparisons.

2 System Design: We analyze and design the details for the individual system components, such as:UI components for file selection, variable selection and buttons for running algorithms.Functions for reading CSV/Excel files and data processing.Integration of categorization and clustering algorithms.

3. ImplementationCoding: Develop the application based on the design. This includes:Developing the UI using the Shiny framework.Implementing the functions for reading and processing data from CSV/Excel files.Integrating the categorization (Decision Tree, Logistic Regression) and clustering (K-Means, Hierarchical Clustering) algorithms.Creating graphs with ggplot2.Internal Testing:Testing the code to ensure that all functions are implemented correctly.

4. Testing (Testing)System Testing: Comprehensive testing of the application to ensure that all functions work correctly and work together smoothly.Testing of file loading and data preview.Testing of variable selection and graph generation.Testing of the accuracy and performance of the categorization and clustering algorithms.Confirming that results and comparisons are displayed correctly.Bug Fixes: Correcting any bugs found during testing.

5. Integration and Maintenance System Integration: We integrate the application into a production environment, ensuring that it is accessible to end users.Monitoring and Support. We provide support to users and collect feedback for improvements.Maintenance: Troubleshooting and improving the application based on user feedback and new requirements as they arise.

Conclusions: Using the Waterfall model to develop the data analytics application allows for a structured and linear approach, which is appropriate when requirements are well defined from the start. This process ensures that each phase is fully completed before moving on to the next, thus ensuring the quality and stability of the application before it is made available to the general public.

10 Github

link : https://github.com/mariampal11/TL-ergasia.git In this link is the github repository. it contains the application code , the LaTeX code , the dockerfile , the docker-compose.yml and a docker-steps file which explains in detail what steps we followed and at the end it has the link of our application.