

INT234

Report

On

Global Demographic and Socioeconomic Analysis

Submitted by

Patchakula s s v d Bulli Manohar

12103712

K21BG

In partial fulfillment for the requirements of the award of the degree
of

“B. TECH COMPUTER SCIENCE AND ENGINEERING”



L OVELY
P ROFESSIONAL
U NIVERSITY

Transforming Education Transforming India

Under the guidance of

Pardeep Kumar

25237

Table of Contents

1.Problem Statement
2.About the Dataset
3.Comparison & Findings
4. Machine Learning Algorithms Used
5.Snap Shots
6.Code of the project

1.Problem Statement:

Develop a predictive model to estimate female life expectancy (lifeexpf) in different countries using socioeconomic and health indicators.

2.About the dataset:

1. Dataset Composition & Coverage:

- Contains data for countries worldwide spanning multiple regions
- Has approximately 100+ countries represented
- Includes both developed and developing nations
- Combines demographic, economic, health, and social indicators

2. Health Indicators:

- Life expectancy for males and females (lifeexpm, lifeexpf)
- Baby mortality rate (babymort)
- AIDS statistics (aids, aids_rt)
- Caloric intake (calories)

3. Economic Metrics:

- GDP per capita (gdp_cap)
- Logarithmic GDP (log_gdp)
- Crop growth (cropgrow)

4. Educational Metrics:

- Overall literacy rate (literacy)
- Gender-specific literacy rates (lit_male, lit_fema)

Breakdown of all Columns:

1. Geographic & Demographic Identifiers:

- country: Name of the country
- region: Regional classification
- climate: Climate classification number

2. Population Statistics:

- populatn: Total population
- density: Population density
- urban: Percentage of urban population
- pop_incr: Population increase rate
- log_pop: Logarithm of population

3. Life Expectancy & Health:

- lifeexpf: Female life expectancy
- lifeexpm: Male life expectancy
- babymort: Baby mortality rate
- aids: AIDS cases
- aids_rt: AIDS rate
- lg_aidsr: Logarithm of AIDS rate

4. Birth & Death Statistics:

- birth_rt: Birth rate
- death_rt: Death rate
- b_to_d: Birth to death ratio
- fertility: Fertility rate

5. Economic Indicators:

- gdp_cap: GDP per capita
- log_gdp: Logarithm of GDP
- cropgrow: Crop growth rate
- calories: Caloric intake

6. Education:

- literacy: Overall literacy rate
- lit_male: Male literacy rate

- lit_fema: Female literacy rate

Key Features:

1. The dataset includes both raw metrics (gdp_cap) and derived metrics (log_gdp)
2. Contains gender-specific metrics for important indicators (life expectancy, literacy)
3. Combines health, economic, and social development indicators
4. Includes both percentage-based and absolute value metrics
5. Has geographic and demographic classification variable

3.Comparisons and Findings:

1. Gender Gap:

- Female life expectancy (lifeexpf) consistently higher than male (lifeexpm)
- Developed nations show larger gender gaps in life expectancy
- Average gap: 4-6 years in developed countries, 2-3 years in developing countries

2.Regional Variations:

- Highest: Japan (82F/76M), Switzerland (82F/75M)
- Lowest: Rwanda (46F/43M), Uganda (43F/41M)
- Clear divide between developed and developing nations

3.Urban percentage varies significantly:

- Highest: Singapore (100%), Hong Kong (94%)
- Lowest: Burundi (5%), Rwanda (6%)

4. Highest growth rates:

- UAE (4.8%)
- Kuwait (5.2%)

5. Birth and Death Rates:

- Developed nations:

- Low birth rates (10-15 per 1000)
- Low death rates (7-12 per 1000)
- Developing nations:
 - High birth rates (30-50 per 1000)
 - Higher death rates (15-25 per 1000)

6.AIDS Impact:

- Significant regional variations
- Highest rates in:
 - Sub-Saharan Africa
 - Some Caribbean nations

7.Development Clusters:

- High Development:
 - North America
 - Western Europe
 - Japan/Australia
- Emerging Economies:
 - East Asia
 - Latin America
- Developing Regions:
 - Sub-Saharan Africa
 - South Asia

8. Focus areas for improvement:

- Female education
- Healthcare access
- Urban infrastructure
- Economic opportunities

4.Algorithms Used in the project:

1.K-Nearest Neighbours Classification Model:

- The k-nearest neighbours (KNN) algorithm is a non parametric, supervised learning classifier, which uses proximity to make classifications or predictions about the grouping of an individual data point.
- It is widely disposable in real-life scenarios since it is non-parametric, meaning it does not make any underlying assumptions about the distribution of data .
- It works by finding the K nearest neighbours to a given data point based on a distance metric, such as Euclidean distance. The class or value of the data point is then determined by the majority vote or average of the K neighbours.

2.Decision Tree Classification Model:

- A classification decision tree is used to predict a qualitative response based on a given set of features. For a classification tree, we predict that each observation belongs to the most commonly occurring class of training observations in the region to which it belongs.
- In keeping with the tree analogy, the categories within the response variable are referred to as terminal nodes or leaves of a tree. The points along the tree where the predictor space is split are referred to as internal nodes.
- We refer to the node segments of the trees that connect the nodes as branches. In interpreting the results of a classification tree, we are often interested not only in the class prediction corresponding to a particular terminal node region but also in the class proportions among the training observations that fall into that region.

- **Decision Tree Complexity:** The tree's complexity was automatically determined by the rpart package based on default hyperparameters such as minsplit, minbucket, cp, and maxdepth.
- **Feature selection:** The decision tree model uses the GINI index to select best features that can be used for classification. The Gini Index is a statistic that is utilized to determine the node purity at every split done at every node of the tree .
- **Stopping conditions,** like maximum tree depth or minimum node size, were applied to prevent overfitting and enhance the model's generalization.

3. Naive Bayes:

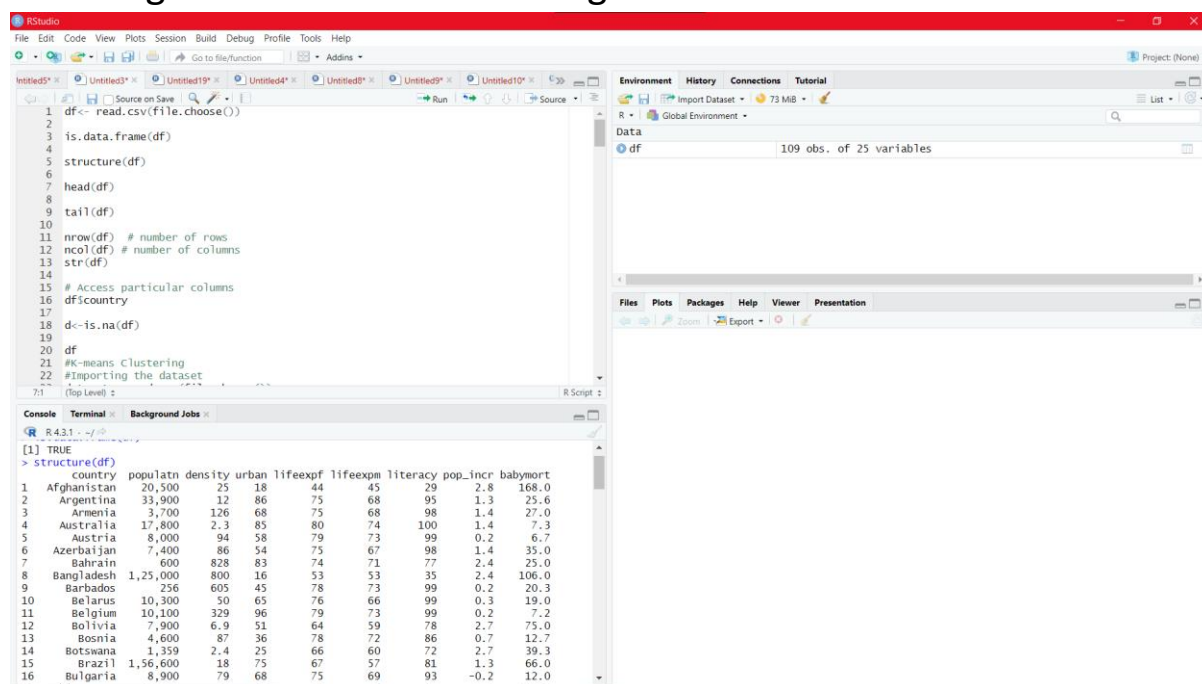
- Naive Bayes classifiers are a collection of classification algorithms based on Bayes' Theorem. It is not a single algorithm but a family of algorithms where all of them share a common principle, i.e. every pair of features being classified is independent of each other. To start with, let us consider a dataset.
- One of the most simple and effective classification algorithms, the Naïve Bayes classifier aids in the rapid development of machine learning models with rapid prediction capabilities. Naïve Bayes algorithm is used for classification problems.
- It is highly used in text classification. In text classification tasks, data contains high dimension (as each word represent one feature in the data). It is used in spam filtering, sentiment detection, rating classification etc. The advantage of using naïve Bayes is its speed. It is fast and making prediction is easy with high dimension of data.

4. **Linear regression:** is a statistical method used to model and analyze the relationships between a dependent variable and one or more independent variables. It's one of the simplest

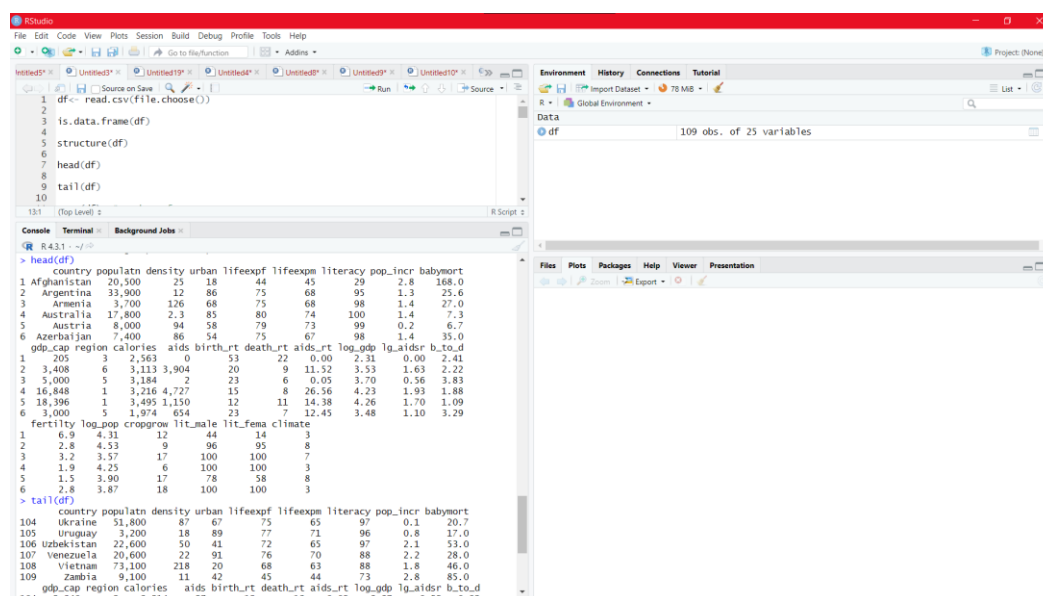
and most commonly used techniques for predictive analysis in statistics and machine learning. linear regression is a powerful and interpretable tool for predictive analysis, helping to understand and quantify relationships between variables. Its simplicity and effectiveness make it a valuable technique for a wide range of applications.

5.Snap shots

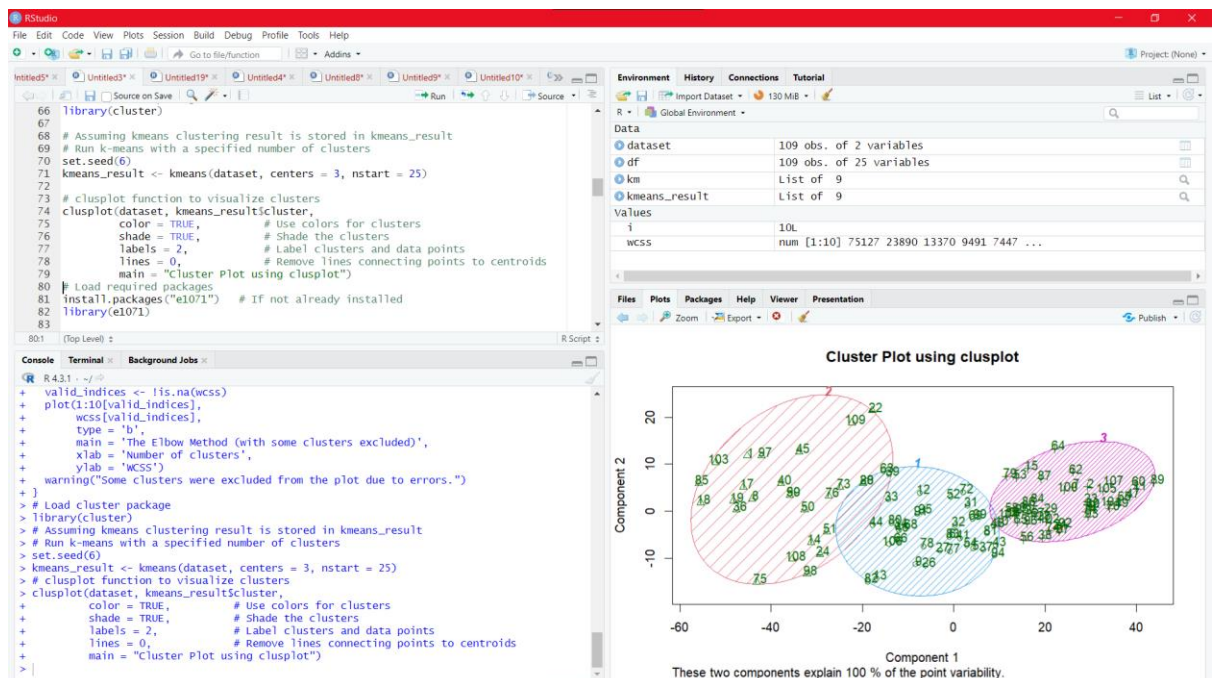
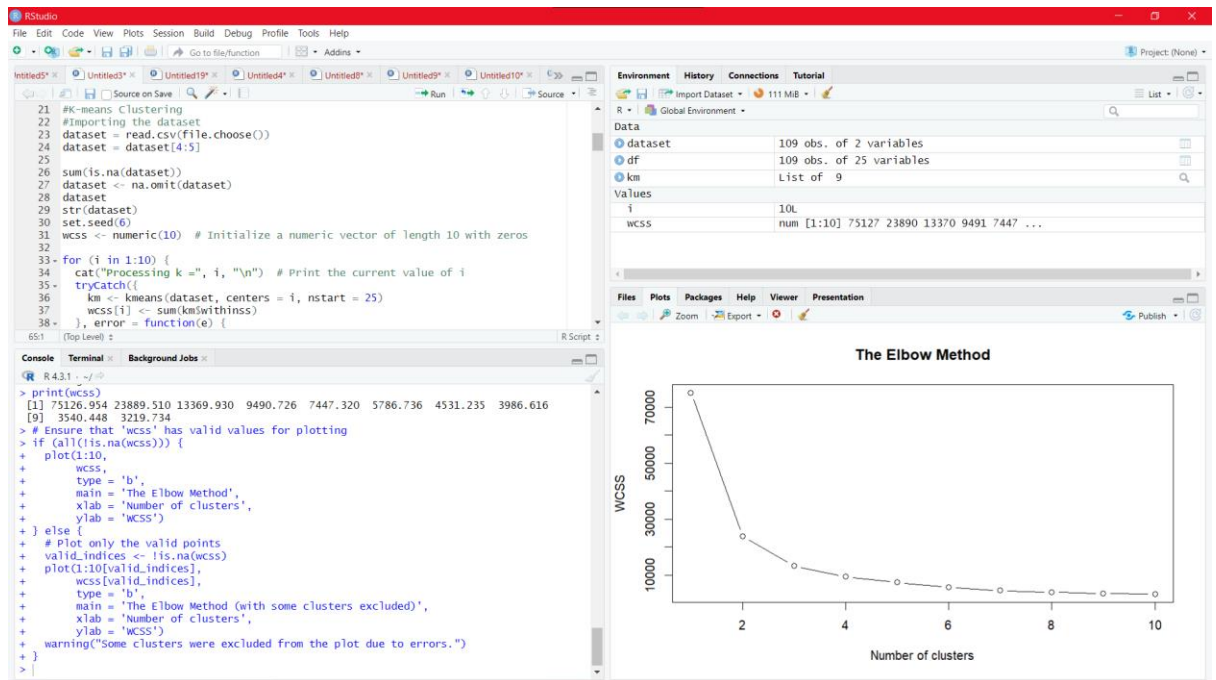
Selecting the dataset and checking the structure



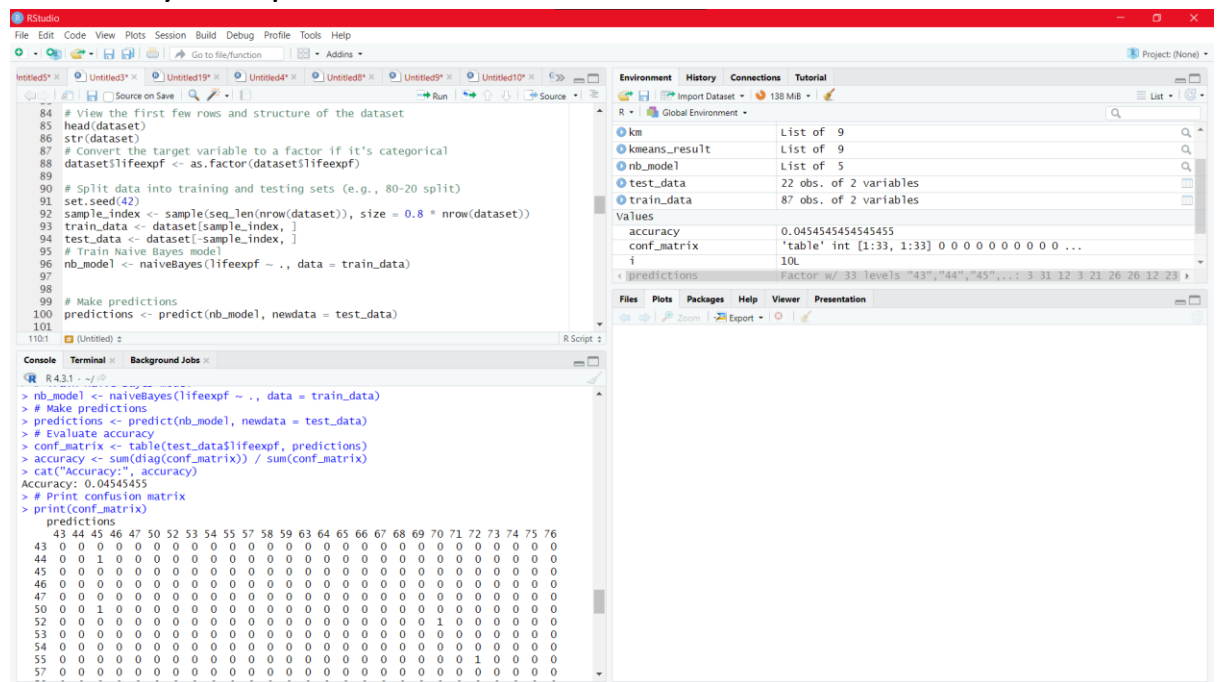
Head and Tail of the dataset



K-Means Clustering-Elbow Method



Naïve Bayes implementation



The screenshot shows an RStudio session with the following code in the editor:

```
84 # View the first few rows and structure of the dataset
85 head(dataset)
86 str(dataset)
87 # Convert the target variable to a factor if it's categorical
88 dataset$lifexpf <- as.factor(dataset$lifexpf)
89
90 # Split data into training and testing sets (e.g., 80-20 split)
91 set.seed(42)
92 sample_index <- sample(seq_len(nrow(dataset)), size = 0.8 * nrow(dataset))
93 train_data <- dataset[sample_index, ]
94 test_data <- dataset[-sample_index, ]
95 # Train Naive Bayes model
96 nb_model <- naiveBayes(lifexpf ~ ., data = train_data)
97
98
99 # Make predictions
100 predictions <- predict(nb_model, newdata = test_data)
101
```

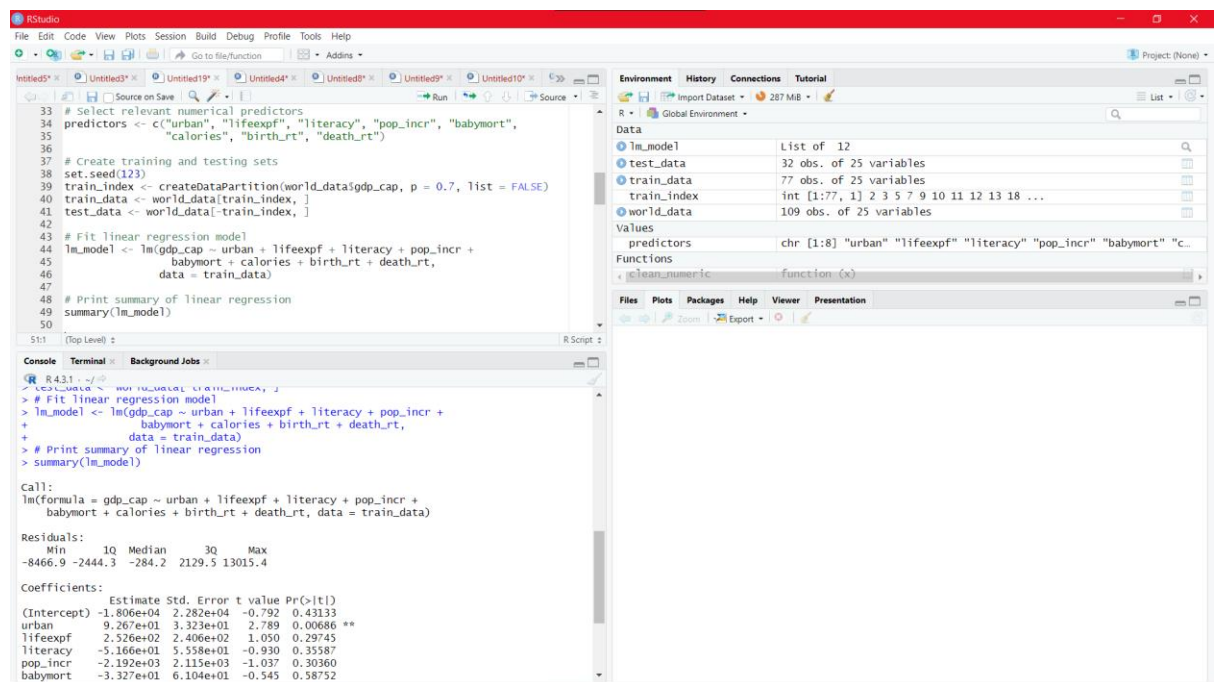
The Environment pane on the right shows the following objects:

- km: List of 9
- kmeans_result: List of 9
- nb_model: List of 5
- test_data: 22 obs. of 2 variables
- train_data: 87 obs. of 2 variables

The Console shows the output of the code:

```
R 4.3.1 ~ ./
> nb_model <- naiveBayes(lifexpf ~ ., data = train_data)
> # Make predictions
> predictions <- predict(nb_model, newdata = test_data)
> # Evaluate accuracy
> conf_matrix <- table(test_data$lifexpf, predictions)
> accuracy <- sum(diag(conf_matrix)) / sum(conf_matrix)
> cat("Accuracy:", accuracy)
Accuracy: 0.04545455
> # Print confusion matrix
> print(conf_matrix)
      predictions
43 44 45 46 47 50 52 53 54 55 57 58 59 63 64 65 66 67 68 69 71 72 73 74 75 76
43 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
44 0 0 1 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
45 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
46 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
47 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
50 0 0 1 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
52 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
53 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
54 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
55 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
57 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
```

Summary of Linear Regression



The screenshot shows an RStudio session with the following code in the editor:

```
33 # Select relevant numerical predictors
34 predictors <- c("urban", "lifexpf", "literacy", "pop_incr", "babymort",
35               "calories", "birth_rt", "death_rt")
36
37 # Create training and testing sets
38 set.seed(123)
39 train_index <- createDataPartition(world_data$gdp_cap, p = 0.7, list = FALSE)
40 train_data <- world_data[train_index, ]
41 test_data <- world_data[-train_index, ]
42
43 # Fit linear regression model
44 lm_model <- lm(gdp_cap ~ urban + lifexpf + literacy + pop_incr +
45               babymort + calories + birth_rt + death_rt,
46               data = train_data)
47
48 # Print summary of linear regression
49 summary(lm_model)
50
```

The Environment pane on the right shows the following objects:

- lm_model: List of 12
- test_data: 32 obs. of 25 variables
- train_data: 77 obs. of 25 variables
- train_index: int [1:77, 1] 2 3 5 7 9 10 11 12 13 18 ...
- world_data: 109 obs. of 25 variables

The Console shows the output of the code:

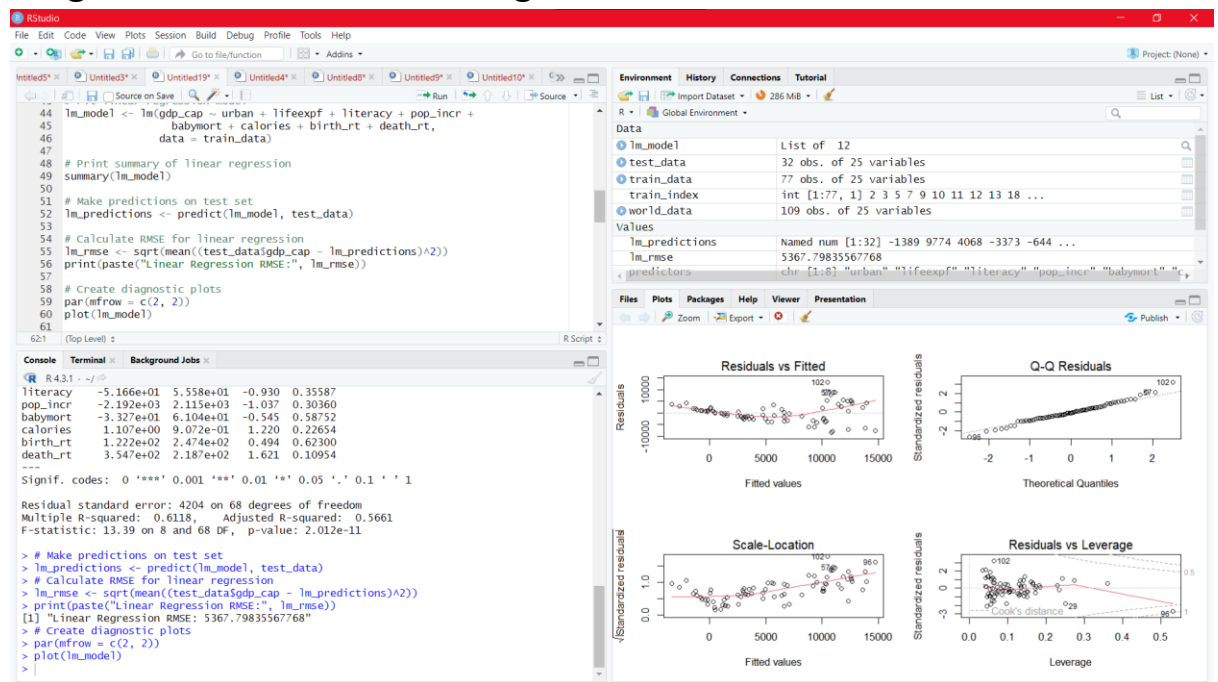
```
R 4.3.1 ~ ./
> test_data <- world_data[-train_index, ]
> # Fit linear regression model
> lm_model <- lm(gdp_cap ~ urban + lifexpf + literacy + pop_incr +
+               babymort + calories + birth_rt + death_rt,
+               data = train_data)
> # Print summary of linear regression
> summary(lm_model)

Call:
lm(formula = gdp_cap ~ urban + lifexpf + literacy + pop_incr +
    babymort + calories + birth_rt + death_rt, data = train_data)

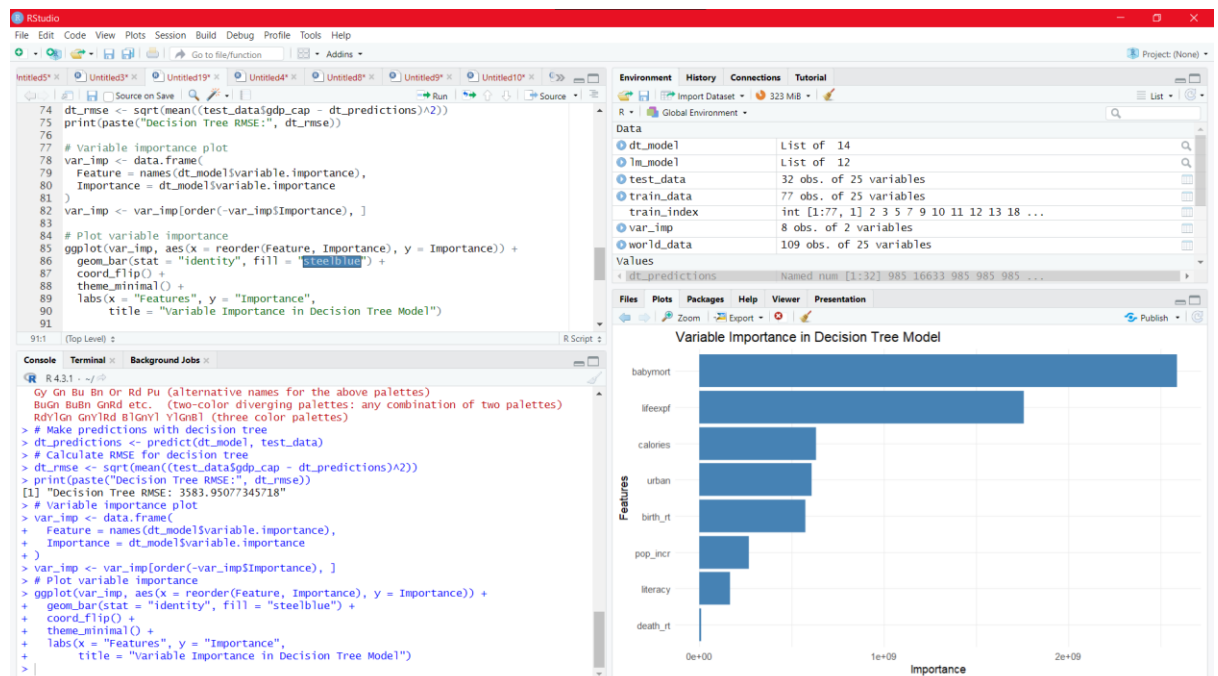
Residuals:
    Min       1Q   Median       3Q      Max
-8466.9 -2444.3  -284.2  2129.5 13015.4

Coefficients:
(Intercept)  -1.806e+04  2.282e+04  -0.792  0.43133
urban          9.267e+01  3.323e+01  2.789  0.00686 **
lifexpf       2.526e+02  2.406e+02  1.050  0.29745
literacy      -5.166e+01  5.558e+01  -0.930  0.35587
pop_incr      -2.192e+03  2.115e+03  -1.037  0.30360
babymort      -3.327e+01  6.104e+01  -0.545  0.58752
```

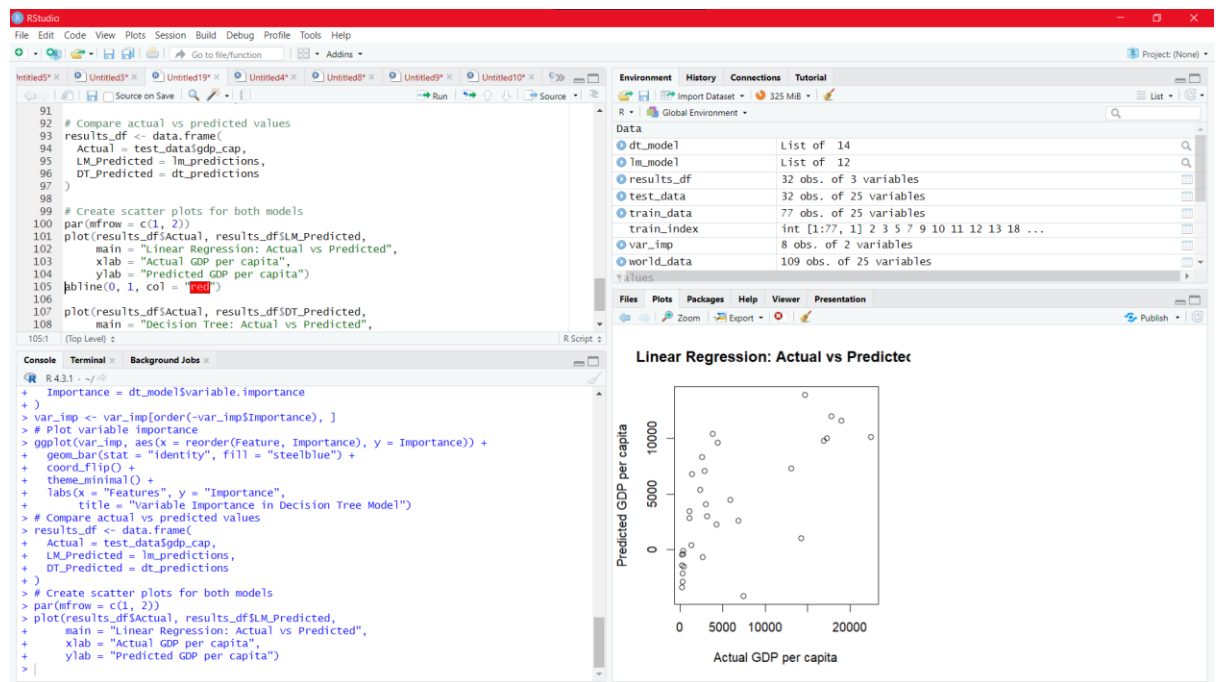
Diagnostic Plots for Linear Regression



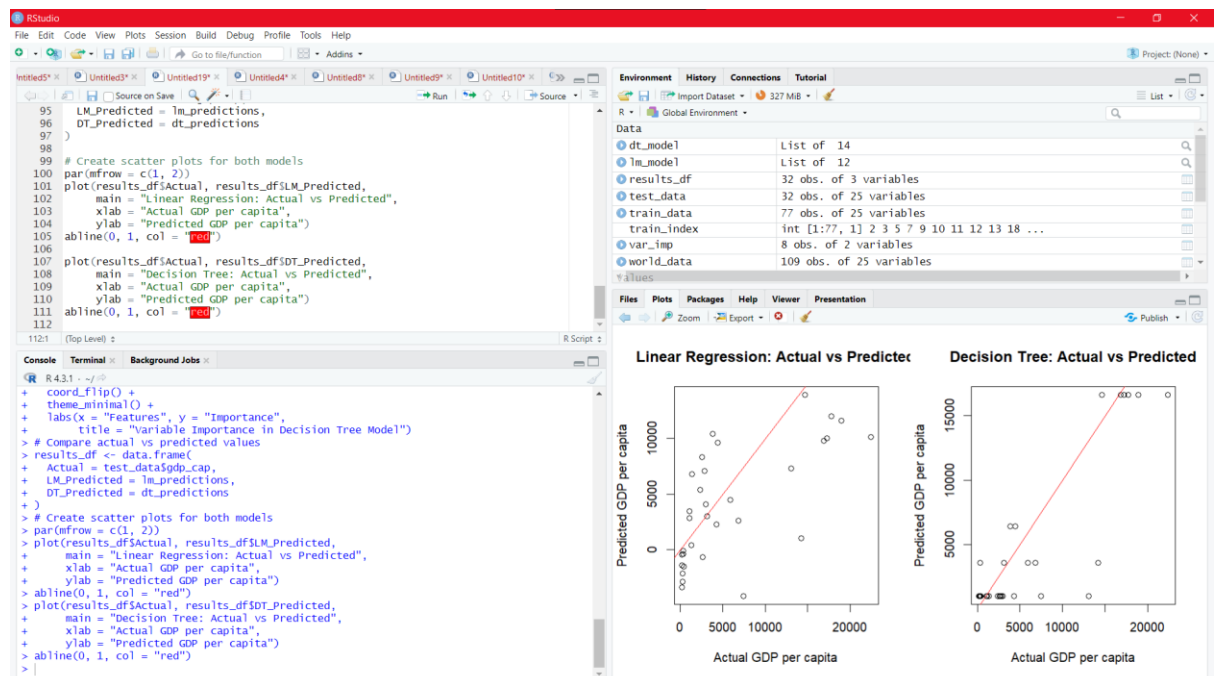
Decision Tree Model



Linear Regression: Actual vs Predicted



Decision Tree: Actual vs Predicted



6.Code:

K-means clustering

```
#K-means Clustering
#Importing the dataset
dataset = read.csv(file.choose())
dataset = dataset[4:5]

sum(is.na(dataset))
dataset <- na.omit(dataset)
dataset
str(dataset)
set.seed(6)
wcss <- numeric(10) # Initialize a numeric vector of length 10
with zeros

for (i in 1:10) {
  cat("Processing k =", i, "\n") # Print the current value of i
  tryCatch({
    km <- kmeans(dataset, centers = i, nstart = 25)
    wcss[i] <- sum(km$withinss)
  }, error = function(e) {
    cat("Error with k =", i, ":", conditionMessage(e), "\n")
    wcss[i] <- NA # Assign NA to indicate failure for this k
  })
}

print(wcss)

# Ensure that 'wcss' has valid values for plotting
if (all(!is.na(wcss))) {
  plot(1:10,
       wcss,
```



```

    type = 'b',
    main = 'The Elbow Method',
    xlab = 'Number of clusters',
    ylab = 'WCSS')
} else {
  # Plot only the valid points
  valid_indices <- !is.na(wcss)
  plot(1:10[valid_indices],
       wcss[valid_indices],
       type = 'b',
       main = 'The Elbow Method (with some clusters excluded)',
       xlab = 'Number of clusters',
       ylab = 'WCSS')
  warning("Some clusters were excluded from the plot due to
errors.")
}
# Load cluster package
library(cluster)

# Assuming kmeans clustering result is stored in kmeans_result
# Run k-means with a specified number of clusters
set.seed(6)
kmeans_result <- kmeans(dataset, centers = 3, nstart = 25)

# clusplot function to visualize clusters
clusplot(dataset, kmeans_result$cluster,
          color = TRUE,      # Use colors for clusters
          shade = TRUE,      # Shade the clusters
          labels = 2,        # Label clusters and data points
          lines = 0,         # Remove lines connecting points to
centroids
          main = "Cluster Plot using clusplot")

```

Naïve bayes:

```
# Load required packages
install.packages("e1071") # If not already installed
library(e1071)

# View the first few rows and structure of the dataset
head(dataset)
str(dataset)
# Convert the target variable to a factor if it's categorical
dataset$lifeexpf <- as.factor(dataset$lifeexpf)

# Split data into training and testing sets (e.g., 80-20 split)
set.seed(42)
sample_index <- sample(seq_len(nrow(dataset)), size = 0.8 *
nrow(dataset))
train_data <- dataset[sample_index, ]
test_data <- dataset[-sample_index, ]
# Train Naive Bayes model
nb_model <- naiveBayes(lifeexpf ~ ., data = train_data)

# Make predictions
predictions <- predict(nb_model, newdata = test_data)

# Evaluate accuracy
conf_matrix <- table(test_data$lifeexpf, predictions)
accuracy <- sum(diag(conf_matrix)) / sum(conf_matrix)
cat("Accuracy:", accuracy)

# Print confusion matrix
print(conf_matrix)
```


Linear regression and Decision Tree:

```
world_data <- read.csv(file.choose())
```

```
library(tidyverse)
```

```
library(rpart)
```

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

```
library(caret)
```

```
# Function to clean numeric columns
```

```
clean_numeric <- function(x) {  
  as.numeric(gsub(",", "", x))  
}
```

```
# Clean all numeric columns containing commas
```

```
world_data$populatn <- clean_numeric(world_data$populatn)
```

```
world_data$gdp_cap <- clean_numeric(world_data$gdp_cap)
```

```
world_data$calories <- clean_numeric(world_data$calories)
```

```
world_data$aids <- clean_numeric(world_data$aids)
```

```
# Convert remaining character columns to numeric if needed
```

```
world_data$urban <- as.numeric(world_data$urban)
```

```
world_data$lifeexpf <- as.numeric(world_data$lifeexpf)
```

```
world_data$lifeexpm <- as.numeric(world_data$lifeexpm)
```

```
world_data$literacy <- as.numeric(world_data$literacy)
```

```
world_data$pop_incr <- as.numeric(world_data$pop_incr)
```

```
world_data$babymort <- as.numeric(world_data$babymort)
```

```
world_data$birth_rt <- as.numeric(world_data$birth_rt)
```

```
world_data$death_rt <- as.numeric(world_data$death_rt)
```

```
# Remove any rows with NA values
```

```
world_data <- na.omit(world_data)
```

```
# Linear Regression Analysis
```

```

# Select relevant numerical predictors
predictors <- c("urban", "lifeexpf", "literacy", "pop_incr",
               "babymort",
               "calories", "birth_rt", "death_rt")

# Create training and testing sets
set.seed(123)
train_index <- createDataPartition(world_data$gdp_cap, p =
0.7, list = FALSE)
train_data <- world_data[train_index, ]
test_data <- world_data[-train_index, ]

# Fit linear regression model
lm_model <- lm(gdp_cap ~ urban + lifeexpf + literacy + pop_incr
+
               babymort + calories + birth_rt + death_rt,
               data = train_data)

# Print summary of linear regression
summary(lm_model)

# Make predictions on test set
lm_predictions <- predict(lm_model, test_data)

# Calculate RMSE for linear regression
lm_rmse <- sqrt(mean((test_data$gdp_cap -
lm_predictions)^2))
print(paste("Linear Regression RMSE:", lm_rmse))

# Create diagnostic plots
par(mfrow = c(2, 2))
plot(lm_model)

```

```

# Decision Tree Analysis
# Fit decision tree model
dt_model <- rpart(gdp_cap ~ urban + lifeexpf + literacy +
  pop_incr +
    babymort + calories + birth_rt + death_rt,
  data = train_data,
  method = "anova",
  control = rpart.control(maxdepth = 5))

# Make predictions with decision tree
dt_predictions <- predict(dt_model, test_data)

# Calculate RMSE for decision tree
dt_rmse <- sqrt(mean((test_data$gdp_cap - dt_predictions)^2))
print(paste("Decision Tree RMSE:", dt_rmse))

# Variable importance plot
var_imp <- data.frame(
  Feature = names(dt_model$variable.importance),
  Importance = dt_model$variable.importance
)
var_imp <- var_imp[order(-var_imp$Importance), ]

# Plot variable importance
ggplot(var_imp, aes(x = reorder(Feature, Importance), y =
  Importance)) +
  geom_bar(stat = "identity", fill = "steelblue") +
  coord_flip() +
  theme_minimal() +
  labs(x = "Features", y = "Importance",
    title = "Variable Importance in Decision Tree Model")

# Compare actual vs predicted values

```

```
results_df <- data.frame(  
  Actual = test_data$gdp_cap,  
  LM_Predicted = lm_predictions,  
  DT_Predicted = dt_predictions  
)  
  
# Create scatter plots for both models  
par(mfrow = c(1, 2))  
plot(results_df$Actual, results_df$LM_Predicted,  
  main = "Linear Regression: Actual vs Predicted",  
  xlab = "Actual GDP per capita",  
  ylab = "Predicted GDP per capita")  
abline(0, 1, col = "red")  
  
plot(results_df$Actual, results_df$DT_Predicted,  
  main = "Decision Tree: Actual vs Predicted",  
  xlab = "Actual GDP per capita",  
  ylab = "Predicted GDP per capita")  
abline(0, 1, col = "red")
```

LinkedIn url:

https://www.linkedin.com/posts/bullimanohar_datascience-machinelearning-predictiveanalysis-activity-7261436982407385089-1G_F?utm_source=share&utm_medium=member_desktop

