

American International University- Bangladesh

Data Science

Final Project Report

Summer 22-23

Name : Nayeem Abdul Qaiyum

ID : 20-43581-1

Section : C

Date Of Submit: 15.08.23

Submitted By

# Abdus Salam

Assistant Professor, CS

Data Set Name: US Births 👶 by Year, State, and Education Level

**Data set Link:** https://www.kaggle.com/datasets/danbraswell/temporary-us-births

**Description:**

This dataset provides birth rates and related data across the 50 states and DC from 2016 to 2021. A particular emphasis is given to detailed information on the mother's educational level.

There are several rows and 9 columns in the data set and they are – State, State.Abbreviation, Year, Gender, Education.Level.of.Mother, Education.Level.Code, Number.of.Births, AverageAge.of.Mother..years., Average.Birth.Weight..g. There are different types of attributes in this dataset and they are integer, numeric, character. Here we apply KNN method to find the highly accurate results.

Table of Contents

1. Import data
2. View the structure of the dataset
3. First few rows of the dataset
4. Column name of the data set
5. Find the type of this dataset column
6. Summary
7. **Data preparation steps**
8. Delete Column (State Abbreviation)
9. **Conversion**
10. Categorical to Numeric (State column)
11. Categorical to Numeric (Gender column)
12. Categorical to Numeric (Education Level of Mother column)
13. **Missing Value**
14. Finding the missing value for all attributes
15. **Normalization**
16. **Correlation**
17. Calculate the correlation between "Education.Level.of.Mother" and "State"
18. Calculate the correlation between "Education.Level.of.Mother" and "Year"
19. Calculate the correlation between "Education.Level.of.Mother" and "Gender"
20. Calculate the correlation between "Education.Level.of.Mother" and "Number.of.Births"
21. Calculate the correlation between "Education.Level.of.Mother" and "Average.Age.of.Mother..years."
22. Calculate the correlation between "Education.Level.of.Mother" and "Average.Birth.Weight..g."
23. **Plot Correlation Matrix**
24. **Training & Testing**
25. **Accuracy**
26. **10-fold cross validation**
27. **Confusion matrix**

**Project Solution**

**Import data:**

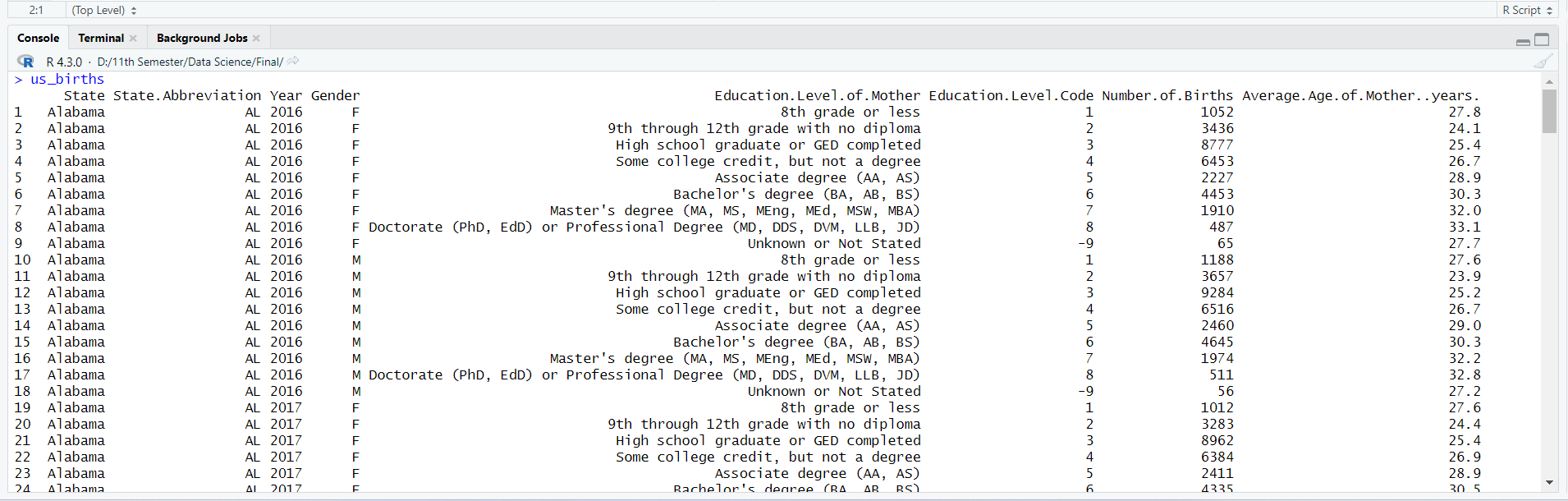
Insert all of the data from the excel file first, and then save the document as a dataset file. then convert the dataset file's format to a CSV file. After importing my CSV file into RStudio, I add the following code.

**Code Segment:**

us\_births <- read.csv("D:/11th Semester/Data Science/Final/us\_births\_2016\_2021.csv")

us\_births

**Output:**

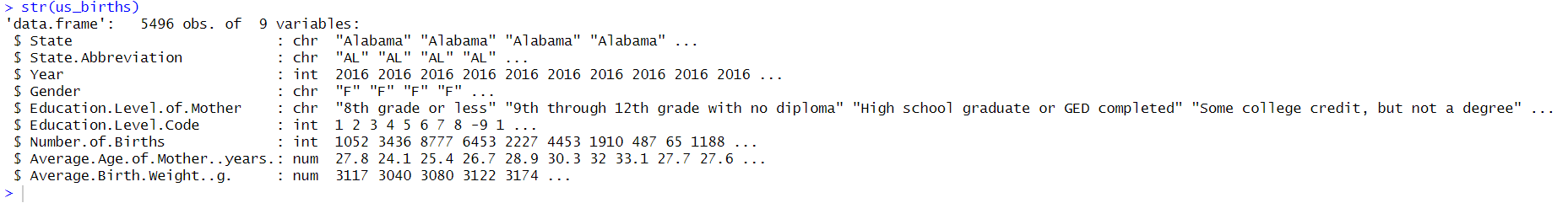
****

**View the structure of the dataset:**

The dataset structure is shown using the str() function, including the variables, their data types, and the initial values. We will get a general idea of the dataset from this.

**Code Segment:**

str(us\_births)

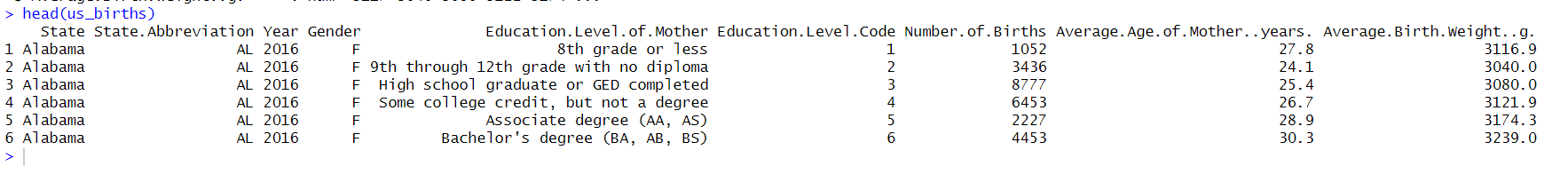


**First few rows of the dataset:**

The first few rows of the dataset are shown using the head() function. This will allow us to understand the data and ensure that it was imported properly.

**Code Segment:**

head(us\_births)

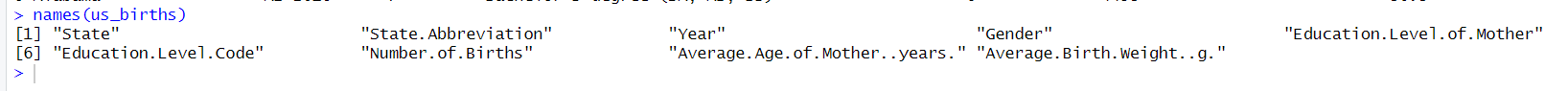


**Column name of the data set:**

**Explanation:** To see the all column name we using the names() function.

**Code Segment:**

names(us\_births)

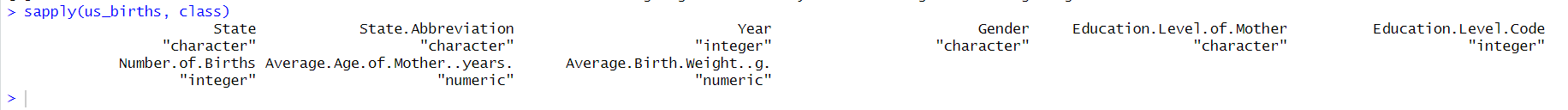
****

**Find the type of this dataset column:**

**Explanation:** We can determine which column contains which type using sapply().

**Code Segment:**

sapply(us\_births, class)

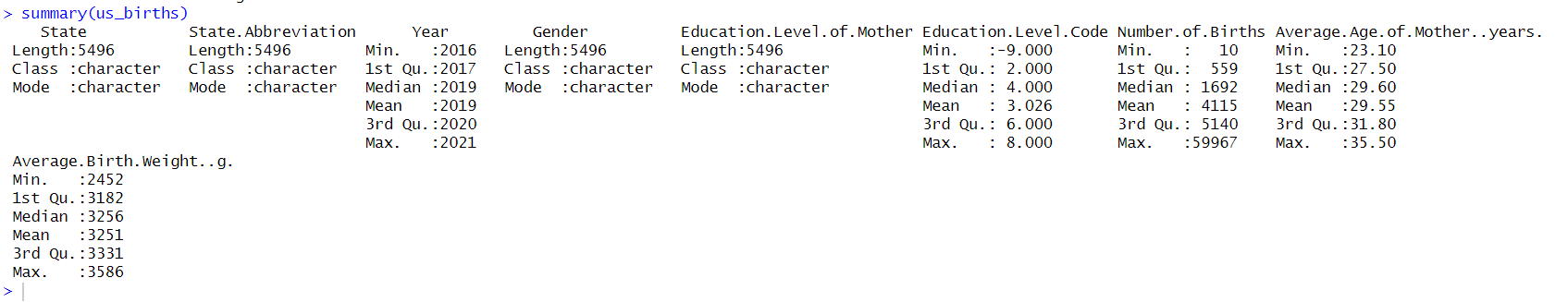
****

**Summary:**

For numerical variables in the dataset, the summary() function returns summary statistics (count, mean, median, etc.). This will help us gain understanding of the variables' distribution and central patterns.

**Code Segment:**

summary(us\_births)



**Data preparation steps**

First, I need to prepare my dataset so that I can apply the KNN method later.

To prepare my dataset firstly I need to convert all categorical data to numerical data. Also, we can delete any column unless we need it.

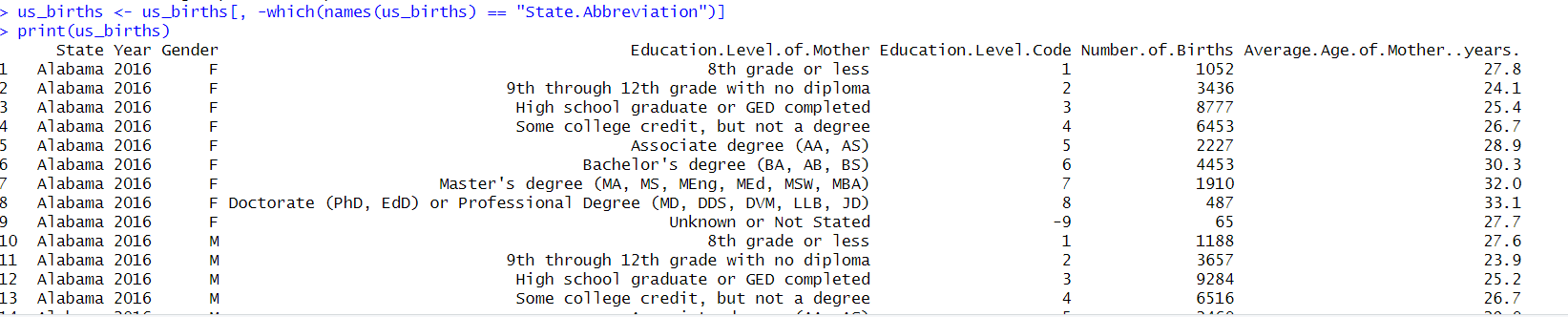
In this dataset I delete one column and that is State Abbreviation.

**Delete Column (State Abbreviation):**

**Code Segment:**

us\_births <- us\_births[, -which(names(us\_births) == "State.Abbreviation")]

print(us\_births)



**Conversion**

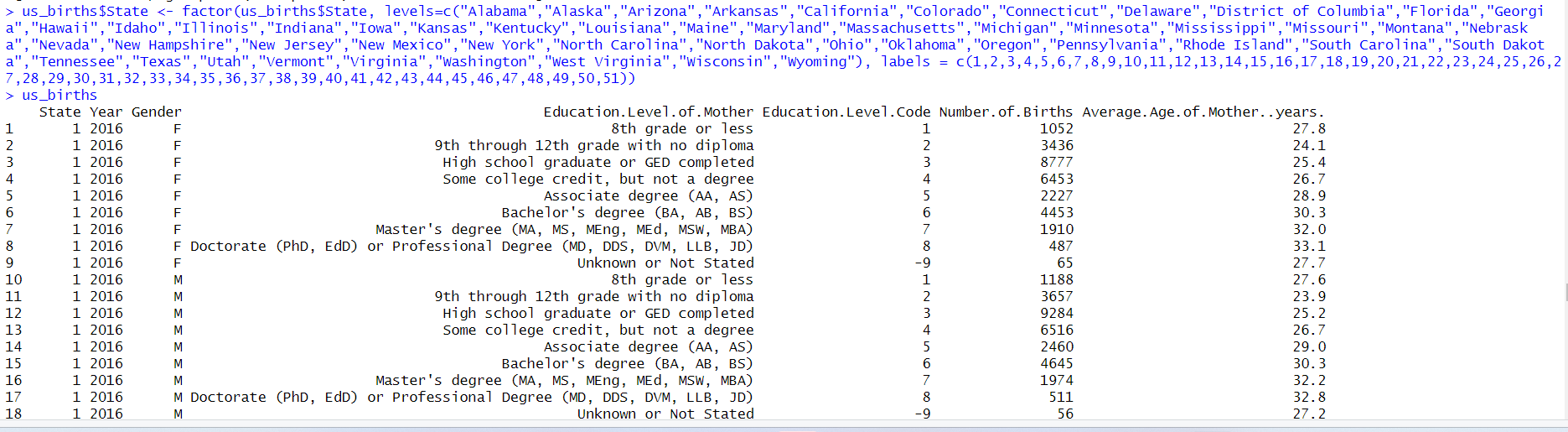
Converting categorical data to numerical data is a common preprocessing step in data science and analysis. This is often necessary because many algorithms, including K-Nearest Neighbors (KNN), work with numerical data and mathematical calculations.

**Categorical to Numeric (State column):**

**Code Segment:**

us\_births$State<factor(us\_births$State,levels=c("Alabama","Alaska","Arizona","Arkansas","California","Colorado","Connecticut","Delaware","District of Columbia","Florida","Georgia","Hawaii","Idaho","Illinois","Indiana","Iowa","Kansas","Kentucky","Louisiana","Maine","Maryland","Massachusetts","Michigan","Minnesota","Mississippi","Missouri","Montana","Nebraska","Nevada","New Hampshire","New Jersey","New Mexico","New York","North Carolina","North Dakota","Ohio","Oklahoma","Oregon","Pennsylvania","Rhode Island","South Carolina","South Dakota","Tennessee","Texas","Utah","Vermont","Virginia","Washington","West Virginia","Wisconsin","Wyoming"), labels = c(1,2,3,4,5,6,7,8,9,10,11,12,13,14,15,16,17,18,19,20,21,22,23,24,25,26,27,28,29,30,31,32,33,34,35,36,37,38,39,40,41,42,43,44,45,46,47,48,49,50,51))

us\_births

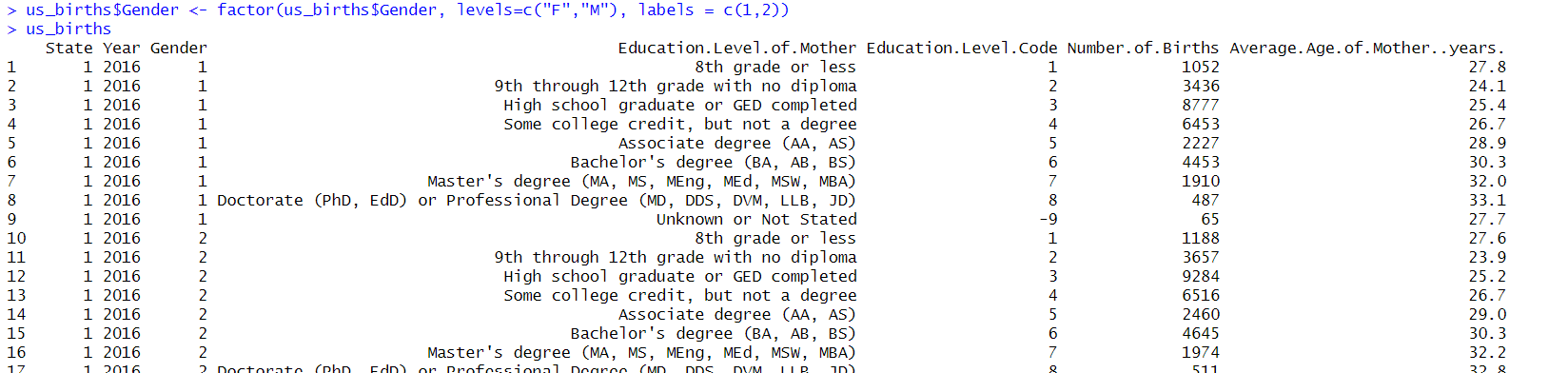


**Categorical to Numeric (Gender column):**

**Code Segment:**

us\_births$Gender <- factor(us\_births$Gender, levels=c("F","M"), labels = c(1,2))

us\_births

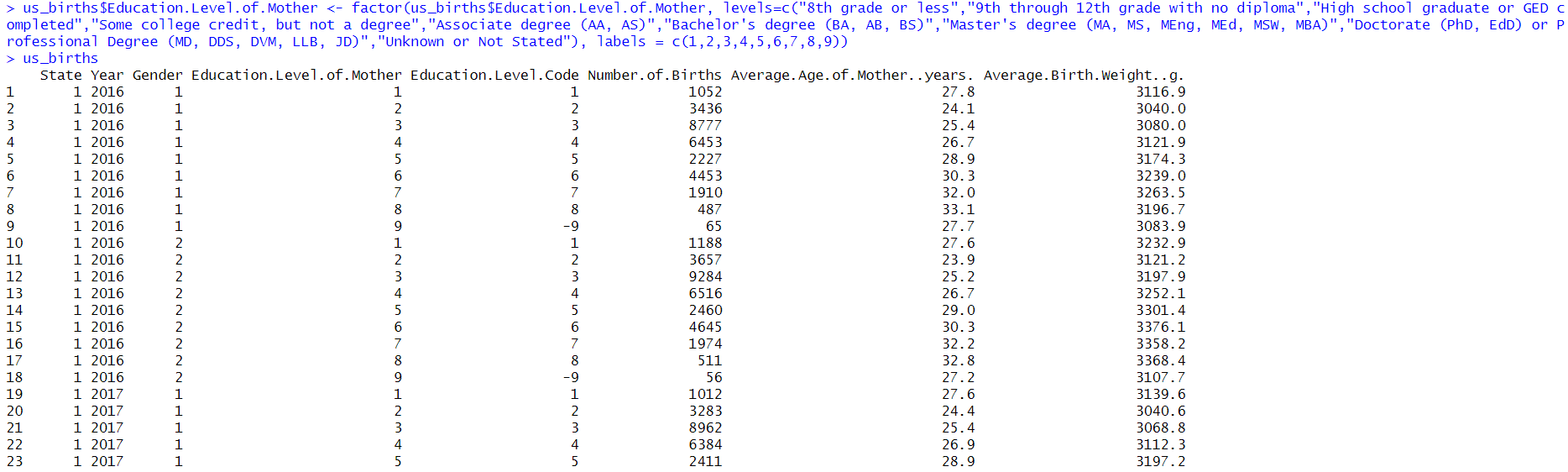


**Categorical to Numeric (Education Level of Mother column):**

**Code Segment:**

us\_births$Education.Level.of.Mother <- factor(us\_births$Education.Level.of.Mother, levels=c("8th grade or less","9th through 12th grade with no diploma","High school graduate or GED completed","Some college credit, but not a degree","Associate degree (AA, AS)","Bachelor's degree (BA, AB, BS)","Master's degree (MA, MS, MEng, MEd, MSW, MBA)","Doctorate (PhD, EdD) or Professional Degree (MD, DDS, DVM, LLB, JD)","Unknown or Not Stated"), labels = c(1,2,3,4,5,6,7,8,9))

us\_births



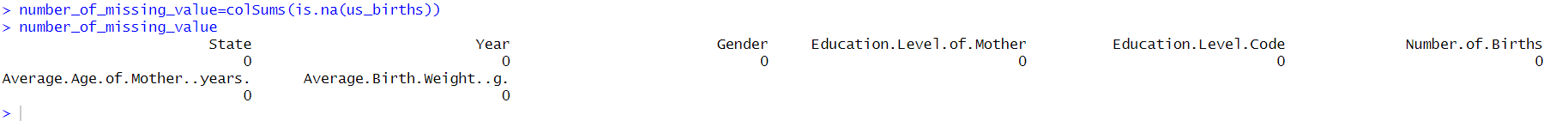
**Finding the missing value for all attributes:**

Missing data is crucial for accurate analysis and results.

**Code Segment:**

number\_of\_missing\_value=colSums(is.na(us\_births))

number\_of\_missing\_value



**Normalization**

Normalization is a data preprocessing technique that is commonly used in data science to scale and transform features to a consistent range (0,1). It involves adjusting the values of features in a dataset to ensure that they have similar scales.

**Code Segment:**

library(dplyr)

us\_births <- as.data.frame(sapply(us\_births, as.numeric))

min\_max\_norm <- function(x) {

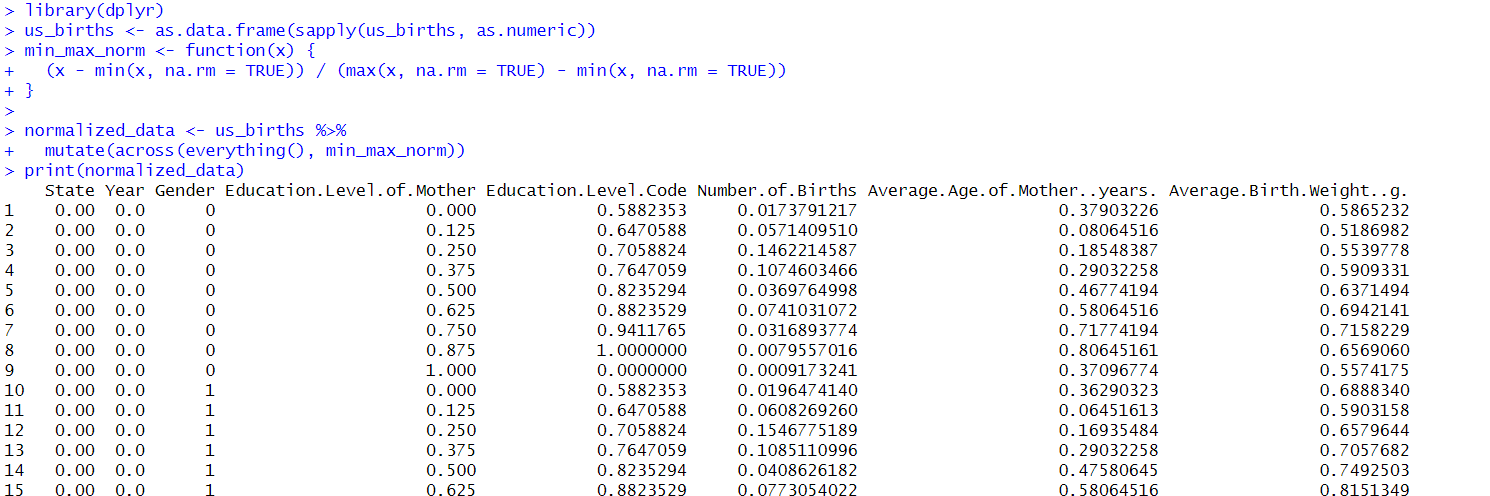
(x - min(x, na.rm = TRUE)) / (max(x, na.rm = TRUE) - min(x, na.rm = TRUE))

}

normalized\_data <- us\_births %>%

mutate(across(everything(), min\_max\_norm))

print(normalized\_data)



**Correlation**

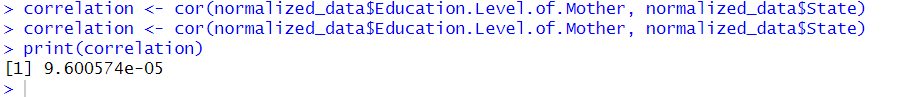
Correlation analysis is a statistical technique used to evaluate the strength and direction of the linear relationship between two or more variables in a dataset.

**Calculate the correlation between "Education.Level.of.Mother" and "State":**

**Code Segment:**

correlation <- cor(normalized\_data$Education.Level.of.Mother, normalized\_data$State)

print(correlation)

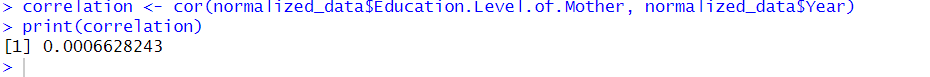


**Calculate the correlation between "Education.Level.of.Mother" and "Year":**

**Code Segment:**

correlation <- cor(normalized\_data$Education.Level.of.Mother, normalized\_data$Year)

print(correlation)

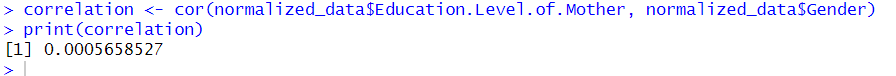
****

**Calculate the correlation between "Education.Level.of.Mother" and "Gender":**

**Code Segment:**

correlation <- cor(normalized\_data$Education.Level.of.Mother, normalized\_data$Gender)

print(correlation)

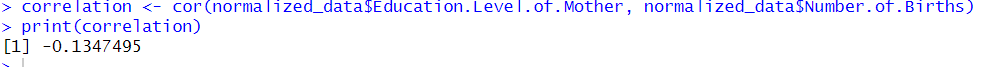
****

**Calculate the correlation between "Education.Level.of.Mother" and "Number.of.Births":**

**Code Segment:**

correlation <- cor(normalized\_data$Education.Level.of.Mother, normalized\_data$Number.of.Births)

print(correlation)

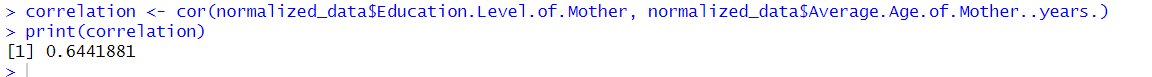
****

**Calculate the correlation between "Education.Level.of.Mother" and "Average.Age.of.Mother..years.":**

**Code Segment:**

correlation <- cor(normalized\_data$Education.Level.of.Mother, normalized\_data$Average.Age.of.Mother..years.)

print(correlation)

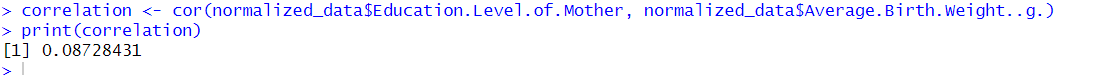
****

**Calculate the correlation between "Education.Level.of.Mother" and "Average.Birth.Weight..g.":**

**Code Segment:**

correlation <- cor(normalized\_data$Education.Level.of.Mother, normalized\_data$Average.Birth.Weight..g.)

print(correlation)

****

**Plot Correlation Matrix**

A plot correlation matrix is a data visualization technique that visually represents relationships between multiple variables in a dataset. It displays correlation coefficients between pairs of variables, with color or shading indicating strength and direction. Each cell in the matrix represents the correlation between two variables, and the color or shading of the cell can be used to convey the strength and direction of the correlation.

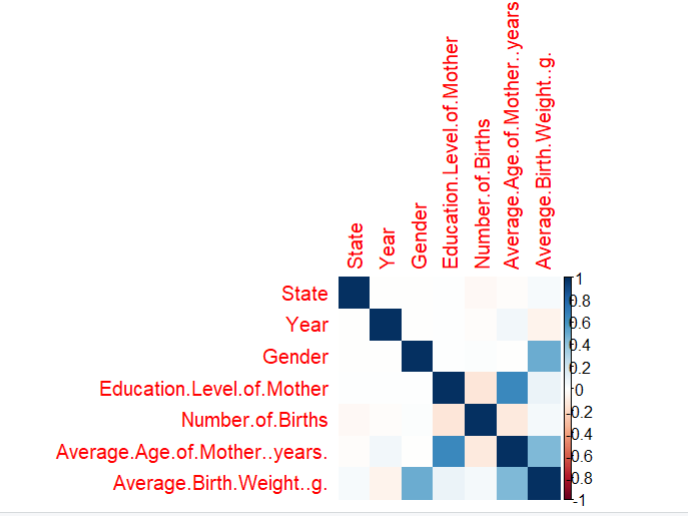
**Code Segment:**

install.packages("corrplot")

library(corrplot)

plot<-cor(normalized\_data)

corrplot(plot,method="color")



**Training & Testing**

Splitting a dataset into training and testing subsets is a crucial step in the field of data science, particularly when building and evaluating predictive models. For example: Fair Comparison, Decision Making, Validation of Results, Quality Control.

Dividing the data into training and test set.

**Code Segment:**

random <- sample(1:nrow(normalized\_data), 0.7 \* nrow(normalized\_data))

# Divide the data set into training and testing sets

Education.Level.of.Mother\_train <- normalized\_data[random, ]

Education.Level.of.Mother\_test <- normalized\_data[-random, ]

# Extract the labels (assuming "Education.Level.of.Mother" column is the label)

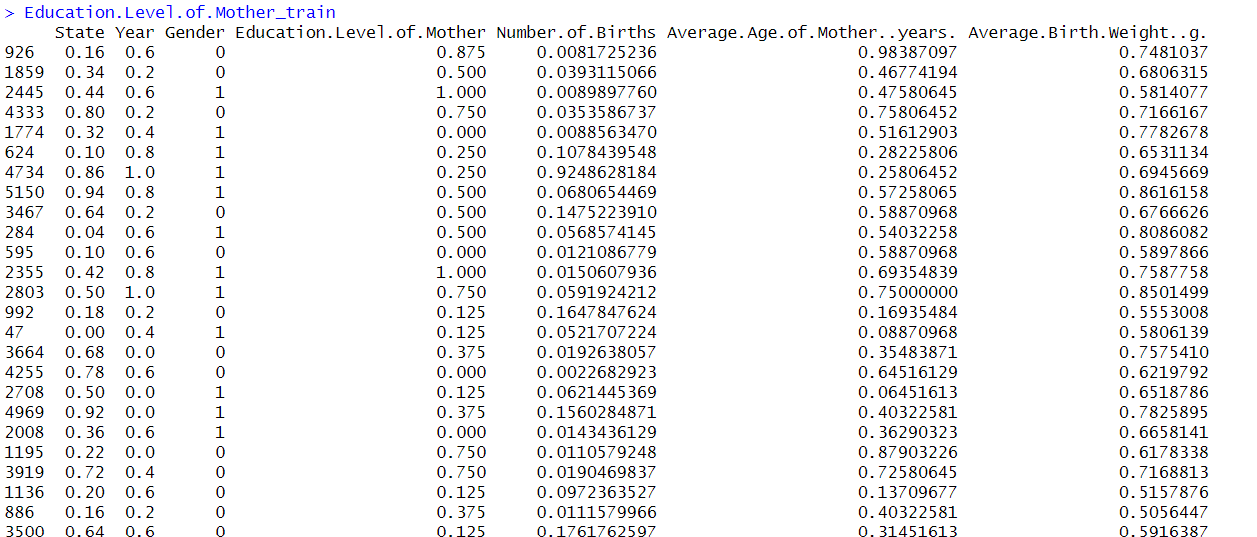
Education.Level.of.Mother\_train\_labels <- Education.Level.of.Mother\_train$Education.Level.of.Mother

Education.Level.of.Mother\_test\_labels <- Education.Level.of.Mother\_test$Education.Level.of.Mother

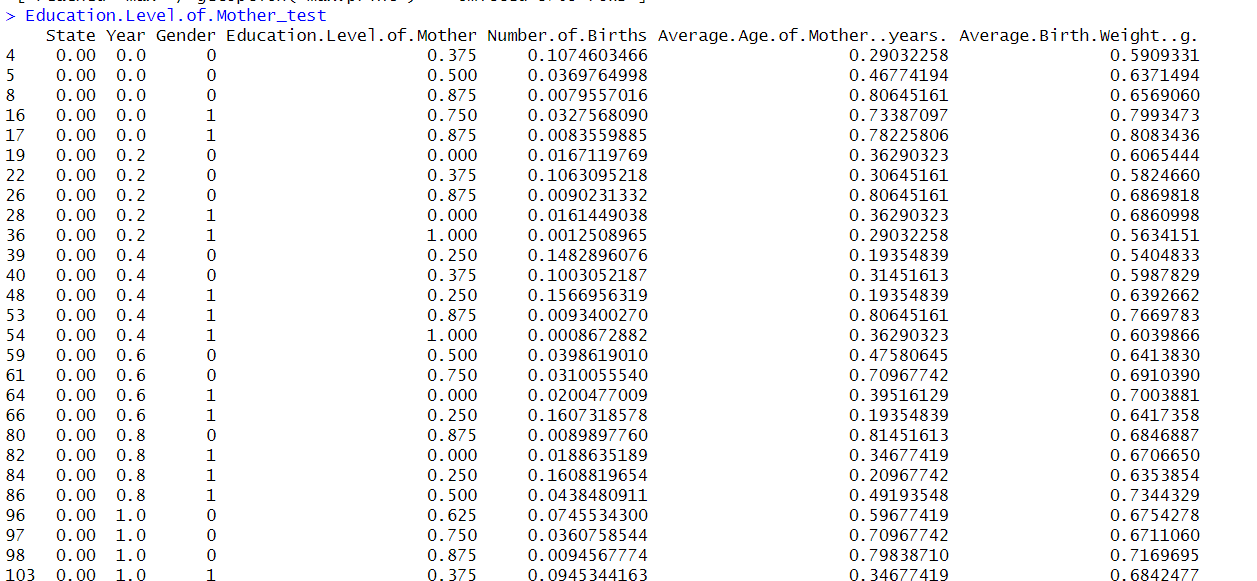
Education.Level.of.Mother\_train

Education.Level.of.Mother\_test

**For train:**



**For test:**

****

**Accuracy:**

In data science and machine learning, accuracy is a key metric used to measure the performance of a predictive model.

**Code Segment:**

library(class)

set.seed(123)

random <- sample(1:nrow(normalized\_data), 0.7 \* nrow(normalized\_data))

Education.Level.of.Mother\_train <- normalized\_data[random, ]

Education.Level.of.Mother\_test <- normalized\_data[-random, ]

Education.Level.of.Mother\_train\_labels <- Education.Level.of.Mother\_train$Education.Level.of.Mother

Education.Level.of.Mother\_test\_labels <- Education.Level.of.Mother\_test$Education.Level.of.Mother

k <- 3

predicted\_labels <- knn(train = Education.Level.of.Mother\_train[, -which(names(Education.Level.of.Mother\_train) == "Education.Level.of.Mother")],

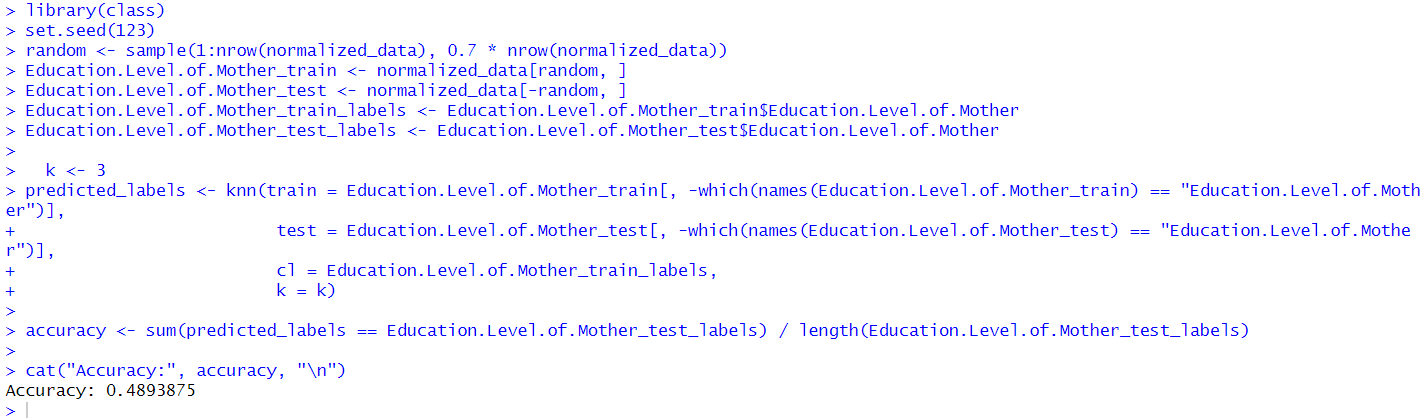
test = Education.Level.of.Mother\_test[, -which(names(Education.Level.of.Mother\_test) == "Education.Level.of.Mother")],

cl = Education.Level.of.Mother\_train\_labels,

k = k)

accuracy <- sum(predicted\_labels == Education.Level.of.Mother\_test\_labels) / length(Education.Level.of.Mother\_test\_labels)

cat("Accuracy:", accuracy, "\n")



**Dividing the data into training and test set**

library(class)

# Split the data into training and test sets

set.seed(123)

random <- sample(1:nrow(normalized\_data), 0.7 \* nrow(normalized\_data))

train\_data <- normalized\_data[random, ]

test\_data <- normalized\_data[-random, ]

# Extract labels

train\_labels <- train\_data$Education.Level.of.Mother

test\_labels <- test\_data$Education.Level.of.Mother

# Define k value

k <- 3

# Train KNN classifier

knn\_model <- knn(train = train\_data[, -which(names(train\_data) == "Education.Level.of.Mother")],

test = test\_data[, -which(names(test\_data) == "Education.Level.of.Mother")],

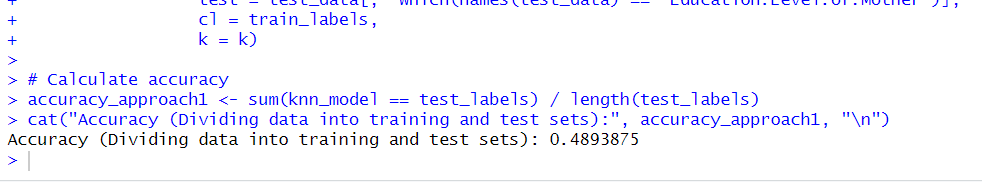
cl = train\_labels,

k = k)

# Calculate accuracy

accuracy\_approach1 <- sum(knn\_model == test\_labels) / length(test\_labels)

cat("Accuracy (Dividing data into training and test sets):", accuracy\_approach1, "\n")



**10-fold cross validation**

The 10-fold cross-validation method, which divides the dataset into 10 equal-sized subsets, is a common data science method for evaluating the effectiveness of predictive models. Its main goal is to give an accurate estimate of how well a model performs on unknown data.

**Code Segment:**

install.packages("class")

install.packages("caret")

library(class)

library(caret)

set.seed(123)

num\_folds <- 10

fold\_indices <- createFolds(normalized\_data$Education.Level.of.Mother, k = num\_folds)

accuracies <- numeric(num\_folds)

for (i in 1:num\_folds) {

test\_indices <- fold\_indices[[i]]

train\_indices <- setdiff(1:nrow(normalized\_data), test\_indices)

Education.Level.of.Mother\_train <- normalized\_data[train\_indices, ]

Education.Level.of.Mother\_test <- normalized\_data[test\_indices, ]

input\_features\_train <- Education.Level.of.Mother\_train[, c("State", "Year", "Gender","Number.of.Births", "Average.Age.of.Mother..years.", "Average.Birth.Weight..g.")]

input\_features\_test <- Education.Level.of.Mother\_test[, c("State", "Year", "Gender","Number.of.Births", "Average.Age.of.Mother..years.", "Average.Birth.Weight..g.")]

Education.Level.of.Mother\_train\_labels <- Education.Level.of.Mother\_train$Education.Level.of.Mother

Education.Level.of.Mother\_test\_labels <- Education.Level.of.Mother\_test$Education.Level.of.Mother

k <- 3 # Set the value of 'k'

predicted\_labels <- knn(train = input\_features\_train,

test = input\_features\_test,

cl = Education.Level.of.Mother\_train\_labels,

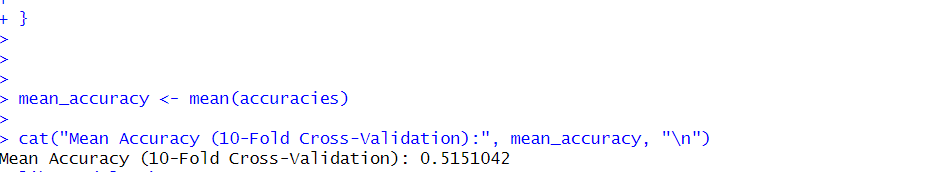
k = k)

accuracies[i] <- sum(predicted\_labels == Education.Level.of.Mother\_test\_labels) / length(Education.Level.of.Mother\_test\_labels)

}

mean\_accuracy <- mean(accuracies)

cat("Mean Accuracy (10-Fold Cross-Validation):", mean\_accuracy, "\n")



**Confusion matrix**

A confusion matrix evaluates classification model performance by comparing predicted and actual classes, revealing strengths and weaknesses, and aiding in data science.

**Code Segment:**

library(class)

library(caret)

# Set seed for reproducibility

set.seed(123)

# Assuming 'normalized\_data' is your original dataset

# Replace this with the correct name if necessary

# Number of folds for cross-validation

num\_folds <- 10

# Create indices for cross-validation folds

fold\_indices <- createFolds(normalized\_data$Education.Level.of.Mother, k = num\_folds)

# Initialize matrices to store confusion matrices and metrics

confusion\_matrices <- list()

recalls <- numeric(num\_folds)

precisions <- numeric(num\_folds)

# Define a function to calculate recall and precision

calculate\_metrics <- function(cm) {

recall <- cm[1, 1] / sum(cm[1, ])

precision <- cm[1, 1] / sum(cm[, 1])

return(list(recall = recall, precision = precision))

}

# Perform 10-fold cross-validation

for (i in 1:num\_folds) {

# Split data into training and testing sets for this fold

test\_indices <- fold\_indices[[i]]

train\_indices <- setdiff(1:nrow(normalized\_data), test\_indices)

data\_train <- normalized\_data[train\_indices, ]

data\_test <- normalized\_data[test\_indices, ]

# Extract the input features and the decision attribute

input\_features\_train <- data\_train[, c("State", "Year", "Gender", "Number.of.Births",

"Average.Age.of.Mother..years.", "Average.Birth.Weight..g.")]

input\_features\_test <- data\_test[, c("State", "Year", "Gender", "Number.of.Births",

"Average.Age.of.Mother..years.", "Average.Birth.Weight..g.")]

decision\_train <- data\_train$Education.Level.of.Mother

decision\_test <- data\_test$Education.Level.of.Mother

# Perform KNN classification

k <- 3 # Set the value of 'k'

predicted\_decisions <- knn(train = input\_features\_train,

test = input\_features\_test,

cl = decision\_train,

k = k)

# Calculate confusion matrix for this fold

confusion\_matrices[[i]] <- table(predicted = predicted\_decisions, actual = decision\_test)

# Calculate recall and precision for this fold

metrics <- calculate\_metrics(confusion\_matrices[[i]])

recalls[i] <- metrics$recall

precisions[i] <- metrics$precision

}

# Calculate the mean recall and precision across folds

mean\_recall <- mean(recalls)

mean\_precision <- mean(precisions)

# Print mean recall and precision

cat("Mean Recall:", mean\_recall, "\n")

cat("Mean Precision:", mean\_precision, "\n")

# Print individual confusion matrices for each fold

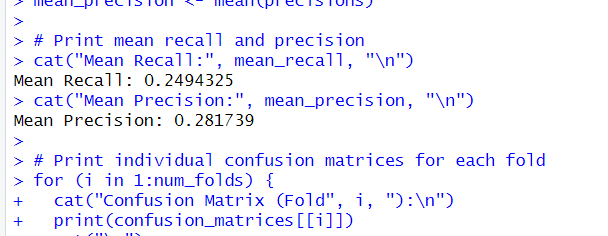
for (i in 1:num\_folds) {

cat("Confusion Matrix (Fold", i, "):\n")

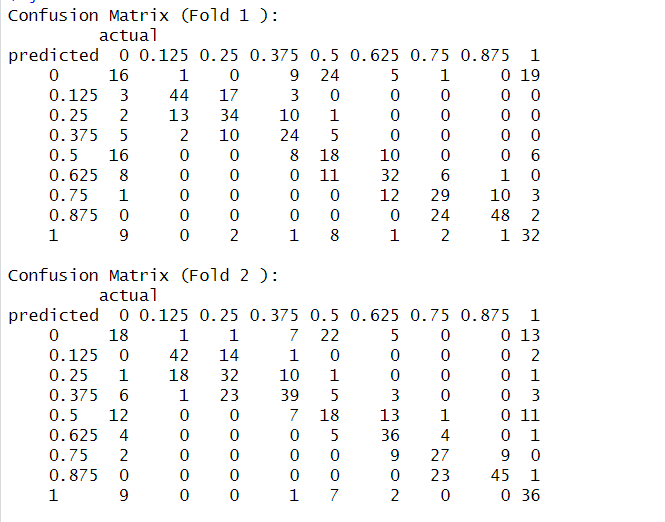
print(confusion\_matrices[[i]])

cat("\n")

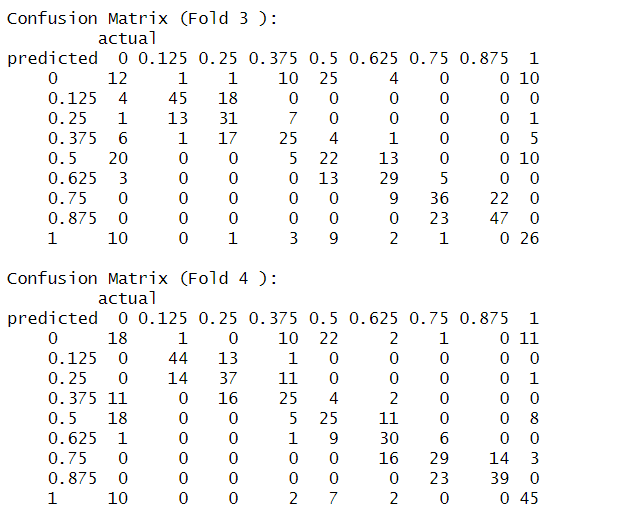
}

****

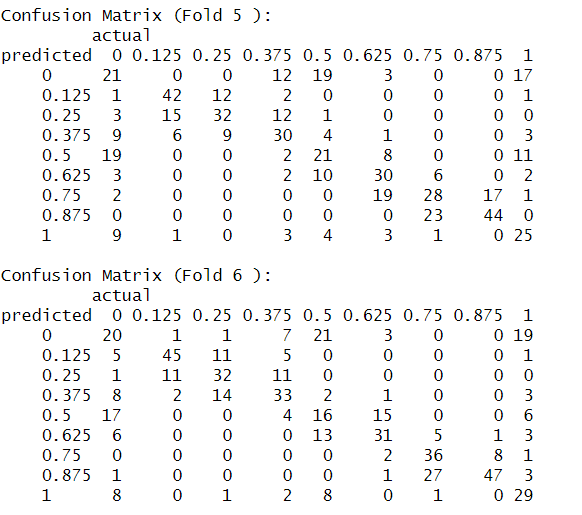
**Confusion Matrix (1,2)**

****

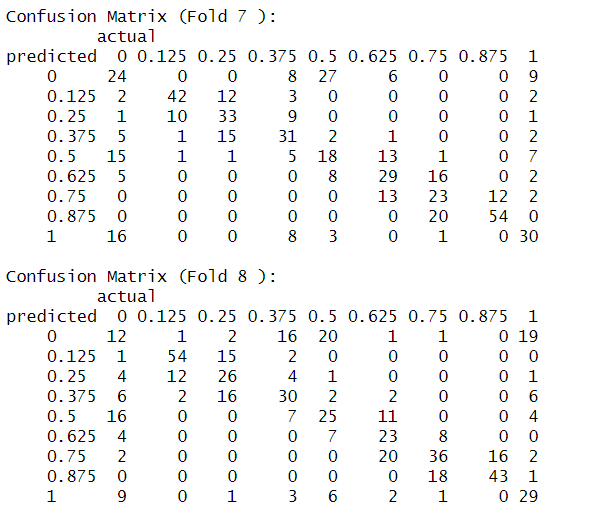
**Confusion Matrix (3,4)**

****

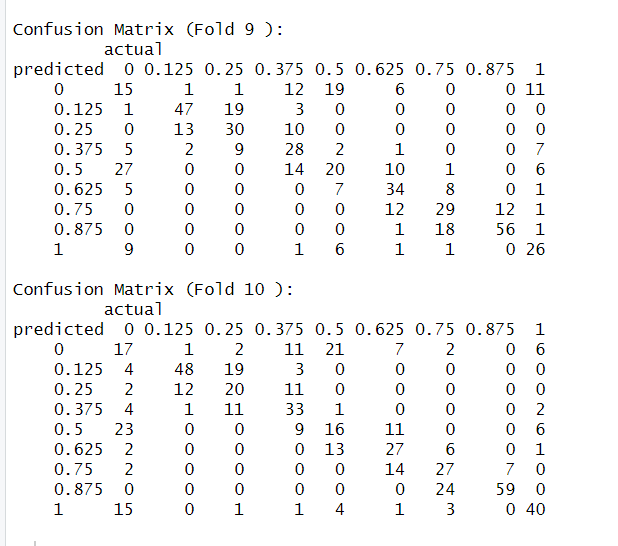
**Confusion Matrix (5,6)**

****

**Confusion Matrix (7,8)**

****

**Confusion Matrix (9,10)**

****