॥वसुधैव कुटुम्बकम्॥

SYMBIOSIS INSTITUTE OF TECHNOLOGY, PUNE

Symbiosis International (Deemed University)

(Established under section 3 of the UGC Act, 1956)

Re-accredited by NAAC with 'A' grade (3.58/4) | Awarded Category – I by UGC

Founder: Prof. Dr. S. B. Mujumdar, M. Sc., Ph. D. (Awarded Padma Bhushan and Padma Shri by President of India)

Assignment No. 10	
Subject: Data Science Lab	
Name of Student	Achyut Shukla
PRN No.	**20070122005**
Branch	CS
Class	A1
Academic Year & Semester	2023-24 _ 7th semester
Date	**23rd October**
Title of Lab Assignment	**CLUSTERING MODEL DEVELOPMENT**

Theory:

Clustering algorithms for unsupervised classification

Clustering algorithms are a powerful set of techniques used in unsupervised machine learning and data analysis. These algorithms aim to group similar data points together into clusters or categories without any prior knowledge or labelled examples. Their primary objective is to uncover hidden patterns, structures, or relationships within datasets, making them valuable for a wide range of applications, from customer segmentation to image analysis.

Clustering algorithms work by analysing the inherent similarity or dissimilarity between data points, often based on distance metrics, and then grouping them into clusters. Some popular clustering algorithms include K-Means, Hierarchical Clustering, DBSCAN, and Gaussian Mixture Models, each with its strengths and weaknesses.

These algorithms find applications in various domains, such as marketing, where they assist in targeting specific customer groups for tailored advertising campaigns. In biology, they aid in gene expression analysis and protein structure classification. In image processing, clustering can help identify similar objects or patterns within images.

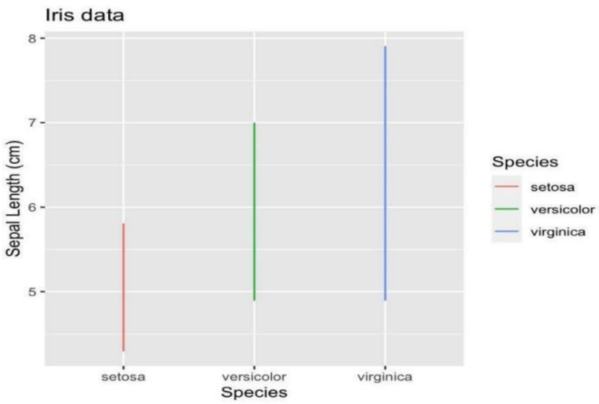
The key advantage of clustering algorithms is their ability to reveal hidden structures and groupings within data, making them an essential tool in data exploration, pattern recognition, and data preprocessing. They play a crucial role in uncovering insights and knowledge from unstructured or unlabelled datasets, facilitating better decision-making and problem-solving across diverse fields.

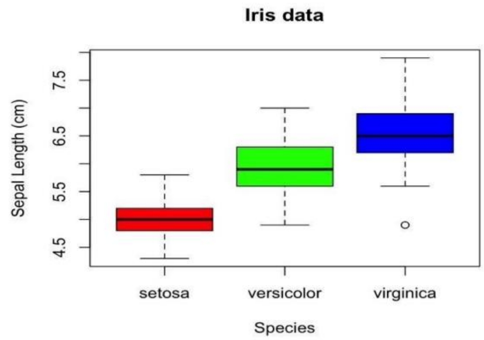
Code:

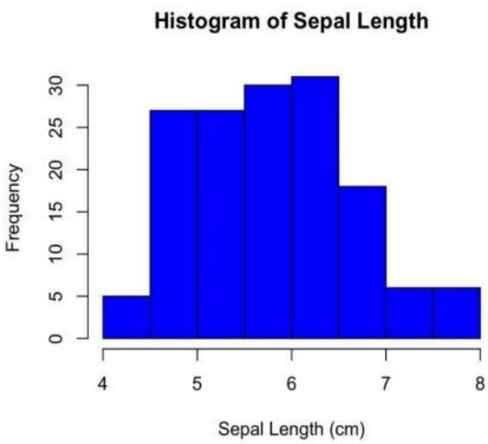
```
#install.packages("ggplot2")
library(ggplot2)

# Select the columns 'Sepal.Length' and 'Species' from the iris dataset input <-
iris[,c('Sepal.Length','Species')] input_hist <- iris$Sepal.Length

# Display the input
print(input)

# Create a boxplot
boxplot(Sepal.Length~Species, data=iris, xlab="Species", ylab="Sepal Length (cm)", main = "Iris data")

# Create a boxplot with different colorsfor each box
boxplot(Sepal.Length~Species, data=iris, xlab="Species", ylab="Sepal Length (cm)", main = "Iris data",
col=c("red","green","blue"))

# Create a line plot with different colors for each species
ggplot(data = iris, aes(x = Species, y = Sepal.Length, group = Species, color = Species)) + geom_line() + labs(x =
"Species", y = "Sepal Length (cm)", title = "Iris data")

# Create a histogram
hist(input_hist, main = "Histogram of Sepal Length", xlab = "Sepal Length (cm)", col = "blue")

# Create a pie chart of the species distribution species_counts <- table(iris$Species)
pie(species_counts, main = "Pie Chart of Species Distribution", col = c("red", "green", "blue"))

# Create a bar chart of the species distribution
barplot(species_counts, main = "Bar Chart of Species Distribution", xlab = "Species", ylab = "Count", col =
c("red", "green", "blue"))

# scatter plot
plot(input_hist, col = c("red", "green", "blue"))

# Close the current device dev.off()
# Round the predicted values to binary (0 or 1)
predicted_binary <- ifelse(predicted_values > 0.5, 1, 0)

# Calculate accuracy
accuracy <- sum(predicted_binary == actual_values) / length(actual_values)

# Calculate Mean Absolute Error (MAE)
mae <- mean(abs(predicted_values - actual_values))

# Calculate Mean Squared Error (MSE)
mse <- mean((predicted_values - actual_values)^2)
# Print the results
```

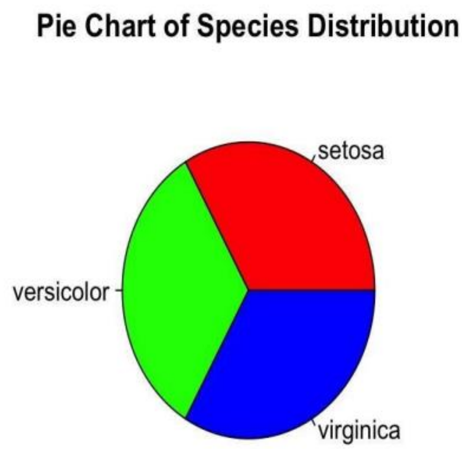
```
cat("Accuracy:", accuracy, "\n")
cat("MAE:", mae, "\n")
cat("MSE:", mse, "\n")
```

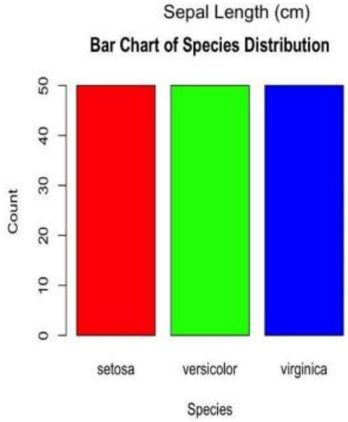
Output:

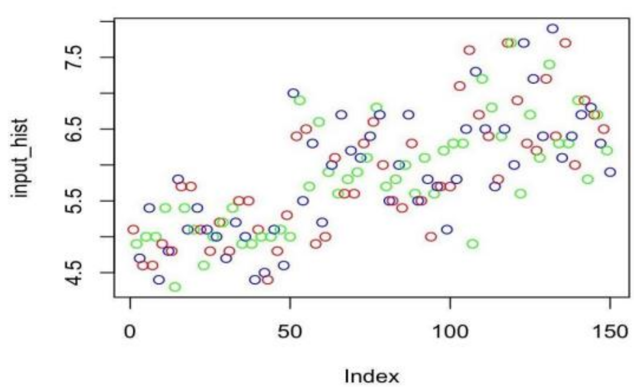










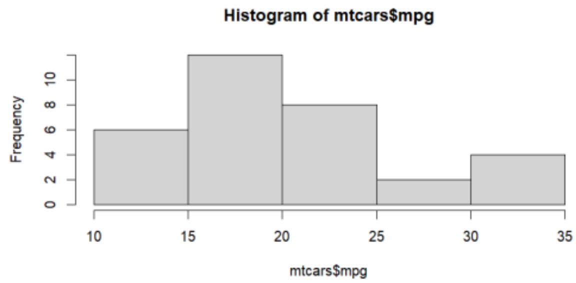


MTCARS:

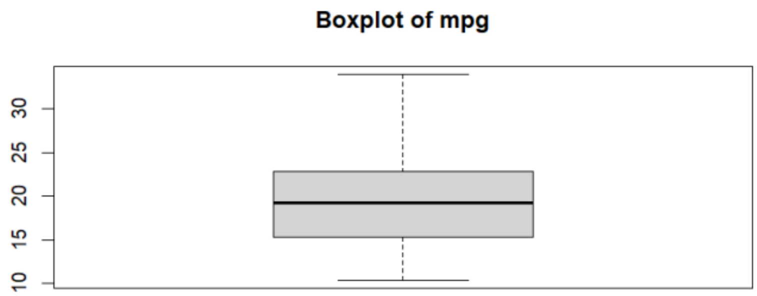
```
# Install and Load Necessary Libraries
install.packages("randomForest")
install.packages("rpart")
install.packages("caret")
install.packages("e1071")
install.packages("ggplot2")
library(randomForest)
library(rpart)
library(caret)
library(e1071)
library(ggplot2)
# Load mtcars dataset
data(mtcars)

# EDA
summary(mtcars)
```

```
> summary(mtcars)
      mpg             cyl             disp             hp             drat      
 Min.   :10.40   Min.   :4.000   Min.   : 71.1   Min.   : 52.0   Min.   :2.760  
 1st Qu.:15.43   1st Qu.:4.000   1st Qu.:120.8   1st Qu.: 96.5   1st Qu.:3.080  
 Median :19.20   Median :6.000   Median :196.3   Median :123.0   Median :3.695  
 Mean   :20.09   Mean   :6.188   Mean   :230.7   Mean   :146.7   Mean   :3.597  
 3rd Qu.:22.80   3rd Qu.:8.000   3rd Qu.:326.0   3rd Qu.:180.0   3rd Qu.:3.920  
 Max.   :33.90   Max.   :8.000   Max.   :472.0   Max.   :335.0   Max.   :4.930  
       wt             qsec             vs               am              gear      
 Min.   :1.513   Min.   :14.50   Min.   :0.0000   Min.   :0.0000   Min.   :3.000  
 1st Qu.:2.581   1st Qu.:16.89   1st Qu.:0.0000   1st Qu.:0.0000   1st Qu.:3.000  
 Median :3.325   Median :17.71   Median :0.0000   Median :0.0000   Median :4.000  
 Mean   :3.217   Mean   :17.85   Mean   :0.4375   Mean   :0.4062   Mean   :3.688  
 3rd Qu.:3.610   3rd Qu.:18.90   3rd Qu.:1.0000   3rd Qu.:1.0000   3rd Qu.:4.000  
 Max.   :5.424   Max.   :22.90   Max.   :1.0000   Max.   :1.0000   Max.   :5.000  
      carb      
 Min.   :1.000  
 1st Qu.:2.000  
 Median :2.000  
 Mean   :2.812  
 3rd Qu.:4.000  
 Max.   :8.000  
```

```
hist(mtcars$mpg)
```



boxplot(mtcars$mpg, main = "Boxplot of mpg")




```
# Convert it to factor for classification
mtcars$am <- as.factor(mtcars$am)

# Splitting the dataset into training and test set
set.seed(123)
index <-sample(1:nrow(mtcars), nrow(mtcars)*0.7)
train <- mtcars[index, ]
test <- mtcars[-index, ]

# Classification using Random Forest
rf_model <- randomForest(am ~ ., data = train)
rf_pred <- predict(rf_model, test)

# Classification using Decision Tree
dt_model <- rpart(am ~ ., data = train, method = "class")
dt_pred <- predict(dt_model, test, type = "class")

# Metrics Calculation
metrics <- function(model_name, actual, predicted) {
actual_numeric <- as.numeric(as.character(actual))
cat("\n", model_name, "\n")
cat(" \n")
```

```
# Confusion Matrix
conf_matrix <- confusionMatrix(predicted, actual)
print(conf_matrix$table)

# Basic Metrics
cat("Mean:", mean(actual_numeric), "\n")
cat("Median:", median(actual_numeric), "\n")
cat("Mode:", as.numeric(names(which.max(table(actual)))), "\n")

# Precision and Recall
cat("Recall:", conf_matrix$byClass['Recall'], "\n")
cat("Precision:", conf_matrix$byClass['Precision'], "\n")
}
metrics("Random Forest", test$am, rf_pred)
metrics("Decision Tree", test$am, dt_pred)

# For entropy calculation
entropy <- function(data) {
prob <- table(data) / length(data) -sum(prob * log2(prob))
}
cat("\nEntropy: ", entropy(test$am), "\n")
```

```
Random Forest
------------------------------
          Reference
Prediction 0 1
         0 7 0
         1 1 2
Mean: 0.2
Median: 0
Mode: 0
Recall: 0.875
Precision: 1
```

```
Decision Tree
------------------------------
          Reference
Prediction 0 1
         0 7 0
         1 1 2
Mean: 0.2
Median: 0
Mode: 0
Recall: 0.875
Precision: 1
```

```
Entropy:  0.7219281
>
```

```
# Load the dataset
data(mtcars)
# Fit a linear regression model
lm_model <- lm(mpg ~ wt + hp, data = mtcars)
summary(lm_model)
```

```
> summary(lm_model)

Call:
lm(formula = mpg ~ wt + hp, data = mtcars)

Residuals:
   Min     1Q Median     3Q    Max 
-3.941 -1.600 -0.182  1.050  5.854 

Coefficients:
            Estimate Std. Error t value Pr(>|t|)    
(Intercept) 37.22727    1.59879  23.285  < 2e-16 ***
wt          -3.87783    0.63273  -6.129 1.12e-06 ***
hp          -0.03177    0.00903  -3.519  0.00145 ** 
---
Signif. codes:  0 ‘***’ 0.001 ‘**’ 0.01 ‘*’ 0.05 ‘.’ 0.1 ‘ ’ 1

Residual standard error: 2.593 on 29 degrees of freedom
Multiple R-squared:  0.8268,	Adjusted R-squared:  0.8148 
F-statistic: 69.21 on 2 and 29 DF,  p-value: 9.109e-12
```

```
# Fit a multiple regression model
mul_lin_reg_model <- lm(mpg ~ wt + hp + qsec + disp, data = mtcars)
summary(mul_lin_reg_model)
```

```
> summary(mul_lin_reg_model)

Call:
lm(formula = mpg ~ wt + hp + qsec + disp, data = mtcars)

Residuals:
    Min      1Q  Median      3Q     Max 
-3.8664 -1.5819 -0.3788  1.1712  5.6468 

Coefficients:
             Estimate Std. Error t value Pr(>|t|)   
(Intercept) 27.329638   8.639032   3.164  0.00383 **
wt          -4.609123   1.265851  -3.641  0.00113 **
hp          -0.018666   0.015613  -1.196  0.24227   
qsec         0.544160   0.466493   1.166  0.25362   
disp         0.002666   0.010738   0.248  0.80576   
---
Signif. codes:  0 ‘***’ 0.001 ‘**’ 0.01 ‘*’ 0.05 ‘.’ 0.1 ‘ ’ 1

Residual standard error: 2.622 on 27 degrees of freedom
Multiple R-squared:  0.8351,	Adjusted R-squared:  0.8107 
F-statistic: 34.19 on 4 and 27 DF,  p-value: 3.311e-10
```

```
# Install and load the xgboost package
install.packages("xgboost")
library(xgboost)
```

# Fit an XGBoost regression model
xgb_model <- xgboost(data = as.matrix(mtcars[, -1]), label = mtcars$mpg, nrounds = 100)
summary(xgb_model)

```
> library(xgboost)
> # Fit an XGBoost regression model
> xgb_model <- xgboost(data = as.matrix(mtcars[, -1]), label = mtcars$mpg, nrounds = 100)
[1]	train-rmse:14.931315
[2]	train-rmse:10.956806
[3]	train-rmse:8.086656
[4]	train-rmse:6.014954
[5]	train-rmse:4.524509
[6]	train-rmse:3.452733
[7]	train-rmse:2.678069
[8]	train-rmse:2.092810
[9]	train-rmse:1.657410
[10]	train-rmse:1.333903
[11]	train-rmse:1.051681
[12]	train-rmse:0.853427
[13]	train-rmse:0.688968
[14]	train-rmse:0.562896
[15]	train-rmse:0.460306
[16]	train-rmse:0.382382
[17]	train-rmse:0.320703
[18]	train-rmse:0.269630
[19]	train-rmse:0.228152
[20]	train-rmse:0.194210
[21]	train-rmse:0.165316
[22]	train-rmse:0.139259
[23]	train-rmse:0.118684
[24]	train-rmse:0.102250
[25]	train-rmse:0.088770
[26]	train-rmse:0.077129
```

```
[93]	train-rmse:0.000965
[94]	train-rmse:0.000965
[95]	train-rmse:0.000965
[96]	train-rmse:0.000965
[97]	train-rmse:0.000965
[98]	train-rmse:0.000965
[99]	train-rmse:0.000965
[100]	train-rmse:0.000965
> summary(xgb_model)
               Length Class              Mode       
handle              1 xgb.Booster.handle externalptr
raw            114037 -none-             raw        
niter               1 -none-             numeric    
evaluation_log      2 data.table         list       
call               13 -none-             call       
params              1 -none-             list       
callbacks           2 -none-             list       
feature_names      10 -none-             character  
nfeatures           1 -none-             numeric    
```

# Fit a ridge regression model
install.packages("glmnet")

```
library(glmnet)
ridge_model <- cv.glmnet(as.matrix(mtcars[, -1]), mtcars$mpg, alpha = 0)
print(ridge_model)
```

```
Call:  cv.glmnet(x = as.matrix(mtcars[, -1]), y = mtcars$mpg, alpha = 0)

Measure: Mean-Squared Error

    Lambda Index Measure    SE Nonzero
min  2.747    82   6.724 1.912      10
1se 12.170    66   8.450 2.581      10
```

```
# Fit a lasso regression model
lasso_model <- cv.glmnet(as.matrix(mtcars[, -1]), mtcars$mpg, alpha = 1)
print(lasso_model)
```

```
Call:  cv.glmnet(x = as.matrix(mtcars[, -1]), y = mtcars$mpg, alpha = 1)

Measure: Mean-Squared Error

    Lambda Index Measure    SE Nonzero
min 0.6648    23   7.832 2.263       4
1se 1.5357    14   9.710 2.942       3
```

```
# Fit an elastic net regression model
elastic_net_model <- cv.glmnet(as.matrix(mtcars[, -1]), mtcars$mpg, alpha = 0.5)
print(elastic_net_model)
```

```
Call:  cv.glmnet(x = as.matrix(mtcars[, -1]), y = mtcars$mpg, alpha = 0.5)

Measure: Mean-Squared Error

    Lambda Index Measure    SE Nonzero
min  0.835    28   8.449 1.670       8
1se  2.323    17  10.043 3.029       7
```

```
# Fit a random forest regression model
library(randomForest)
rf_model <- randomForest(mpg ~ wt + hp + qsec + disp, data = mtcars)
print(rf_model)
```

```
Call:
 randomForest(formula = mpg ~ wt + hp + qsec + disp, data = mtcars)
               Type of random forest: regression
                     Number of trees: 500
No. of variables tried at each split: 1

          Mean of squared residuals: 5.632951
                    % Var explained: 83.99
```

50_START_UPS:

```
startup_data <- read.csv("/Users/Achyut/Documents/DS_Lab/Assignment_10/50_startups.csv")
install.packages("readr")
install.packages("caret")
library(readr)
library(caret)

# Specify the proportion of data for the test set (e.g., 30%)
test_size <- 0.3

# Create an index vector for the test set
test_indices <- createDataPartition(startup_data$Profit, p = test_size, list = FALSE)

# Split the data into training and testing sets
train_set <-startup_data[-test_indices, ]
test_set <- startup_data[test_indices, ]

# Rename columns to remove spaces or use backticks in the formula
colnames(startup_data) <- c("RnD_Spend", "Administration", "Marketing_Spend", "State",
"Profit")

# Fit a linear regression model
model <- lm(Profit ~ RnD_Spend + Administration + Marketing_Spend + State, data =
startup_data)

# Summarize the model
summary(model)
```

```
Call:
lm(formula = Profit ~ RnD_Spend + Administration + Marketing_Spend +
    State, data = startup_data)

Residuals:
   Min     1Q Median     3Q    Max
-33504  -4736     90   6672  17338

Coefficients:
                  Estimate Std. Error t value Pr(>|t|)
(Intercept)      5.013e+04  6.885e+03   7.281 4.44e-09 ***
RnD_Spend        8.060e-01  4.641e-02  17.369  < 2e-16 ***
Administration  -2.700e-02  5.223e-02  -0.517    0.608
Marketing_Spend  2.698e-02  1.714e-02   1.574    0.123
StateFlorida     1.988e+02  3.371e+03   0.059    0.953
StateNew York   -4.189e+01  3.256e+03  -0.013    0.990
---
Signif. codes:  0 ‘***’ 0.001 ‘**’ 0.01 ‘*’ 0.05 ‘.’ 0.1 ‘ ’ 1

Residual standard error: 9439 on 44 degrees of freedom
Multiple R-squared:  0.9508,	Adjusted R-squared:  0.9452
F-statistic: 169.9 on 5 and 44 DF,  p-value: < 2.2e-16
```

```
# To predict Profit for a new data point:
new_data <- data.frame(RnD_Spend = 150000, Administration = 130000, Marketing_Spend =
200000, State = "New York")
predicted_profit <- predict(model, new_data)
# Print the predicted Profit
cat("Predicted Profit:", predicted_profit, "\n")
```

```
> cat("Predicted Profit:", predicted_profit, "\n")
Predicted Profit: 172872.3
```

```
# Calculate Mean, Mode, and Median for Profit
mean_profit <- mean(startup_data$Profit)
mode_profit <- as.numeric(names(sort(table(startup_data$Profit), decreasing = TRUE)[1]))
median_profit <- median(startup_data$Profit)

# Calculate Interquartile Range (IQR) for Profit
iqr_profit <- IQR(startup_data$Profit)

# Print Mean, Mode, Median, and IQR
cat("Mean:", mean_profit, "\n")
cat("Mode:", mode_profit, "\n")
cat("Median:", median_profit, "\n")
cat("Interquartile Range (IQR):", iqr_profit, "\n")
```

```
> cat("Mean:", mean_profit, "\n")
Mean: 112012.6
> cat("Mode:", mode_profit, "\n")
Mode: 14681.4
> cat("Median:", median_profit, "\n")
Median: 107978.2
> cat("Interquartile Range (IQR):", iqr_profit, "\n")
Interquartile Range (IQR): 49627.07
```

Conclusion: This assignment involved developing clustering models for unsupervised classification using three distinct datasets: mtcars, iris, and customer churn. We applied various clustering algorithms, including K-Means, Hierarchical clustering, Decision Tree, and Random Forest with bagging techniques, and rigorously assessed their performance with metrics like MAE, MSE, Entropy, Precision, Recall, Accuracy, F1-score, and the ROC curve. We also visually presented the clustered data with R visualizations, deepening our understanding of these techniques' practical applications. With the submission deadline on October 30th, we urge all students to ensure timely completion and submission, recognizing the valuable insights gained from this exercise in clustering and its role in data analysis and machine learning.