EXPERIMENT:11

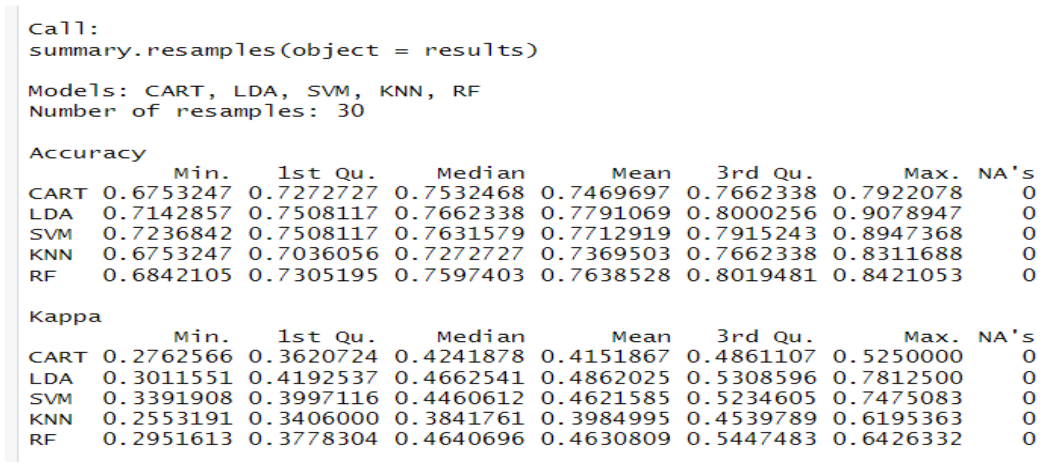
AIM: Create a R program to implement CART, K-Nearest Neighbours, SVM, LDA, Random Forest on PimaIndianDiabetes dataset and compare and visualize the results

DESCRIPTION: The "**PimaIndianDiabetes"** Dataset, originally from the National Institute of Diabetes and Digestive and Kidney Diseases, contains information of 768 women from a population near Phoenix, Arizona, USA. The outcome tested was Diabetes, 258 tested positive and 500 tested negative. Therefore, there is one target (dependent) variable and the 8 attributes (TYNECKI, 2018): pregnancies, OGTT(Oral Glucose Tolerance Test), blood pressure, skin thickness, insulin, BMI(Body Mass Index), age, pedigree diabetes function.

CODE:

```
# load libraries
library(mlbench)
library(caret)
# load the dataset
data(PimaIndiansDiabetes)
# prepare training scheme
control <- trainControl(method="repeatedcv", number=10, repeats=3)
# CART
set.seed(7)
fit.cart<- train(diabetes~., data=PimaIndiansDiabetes, method="rpart", trControl=control)
# LDA
set.seed(7)
fit.lda<- train(diabetes~., data=PimaIndiansDiabetes, method="lda", trControl=control)
# SVM
set.seed(7)
fit.svm<- train(diabetes~., data=PimaIndiansDiabetes, method="svmRadial", trControl=control)
# kNN
set.seed(7)
fit.knn<- train(diabetes~., data=PimaIndiansDiabetes, method="knn", trControl=control)
# Random Forest
set.seed(7)
fit.rf<- train(diabetes~., data=PimaIndiansDiabetes, method="rf", trControl=control)
 # collect resamples
results <- resamples(list(CART=fit.cart, LDA=fit.lda, SVM=fit.svm, KNN=fit.knn, RF=fit.rf))
# summarize differences between modes
summary(results)
```

OUTPUT:



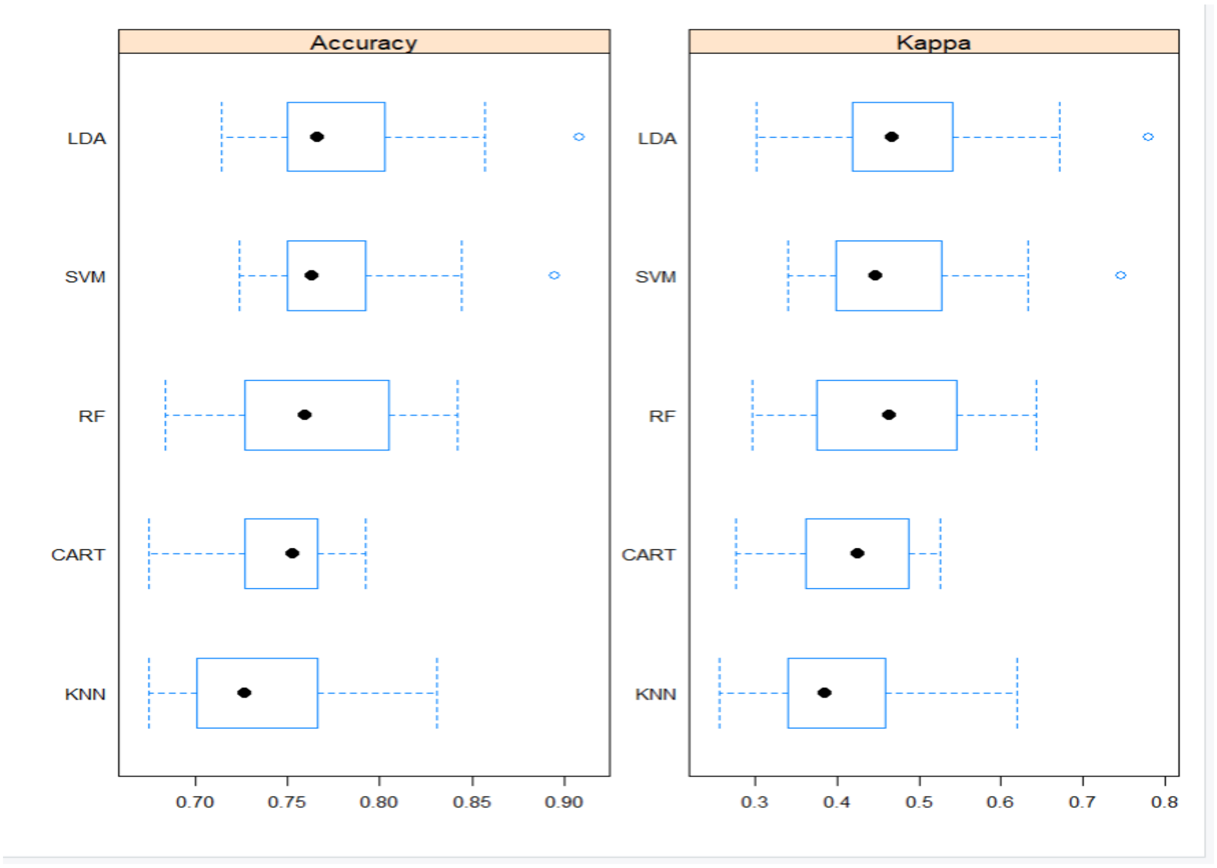
```
Call:
summary.resamples(object = results)

Models: CART, LDA, SVM, KNN, RF
Number of resamples: 30

Accuracy
          Min.   1st Qu.    Median      Mean   3rd Qu.      Max. NA's
CART 0.6753247 0.7272727 0.7532468 0.7469697 0.7662338 0.7922078    0
LDA  0.7142857 0.7508117 0.7662338 0.7791069 0.8000256 0.9078947    0
SVM  0.7236842 0.7508117 0.7631579 0.7712919 0.7915243 0.8947368    0
KNN  0.6753247 0.7036056 0.7272727 0.7369503 0.7662338 0.8311688    0
RF   0.6842105 0.7305195 0.7597403 0.7638528 0.8019481 0.8421053    0

Kappa
          Min.   1st Qu.    Median      Mean   3rd Qu.      Max. NA's
CART 0.2762566 0.3620724 0.4241878 0.4151867 0.4861107 0.5250000    0
LDA  0.3011551 0.4192537 0.4662541 0.4862025 0.5308596 0.7812500    0
SVM  0.3391908 0.3997116 0.4460612 0.4621585 0.5234605 0.7475083    0
KNN  0.2553191 0.3406000 0.3841761 0.3984995 0.4539789 0.6195363    0
RF   0.2951613 0.3778304 0.4640696 0.4630809 0.5447483 0.6426332    0
```

CODE:

# box and whisker plots to compare models

```
scales <- list(x=list(relation="free"), y=list(relation="free"))
bwplot(results, scales=scales)
```

OUTPUT:

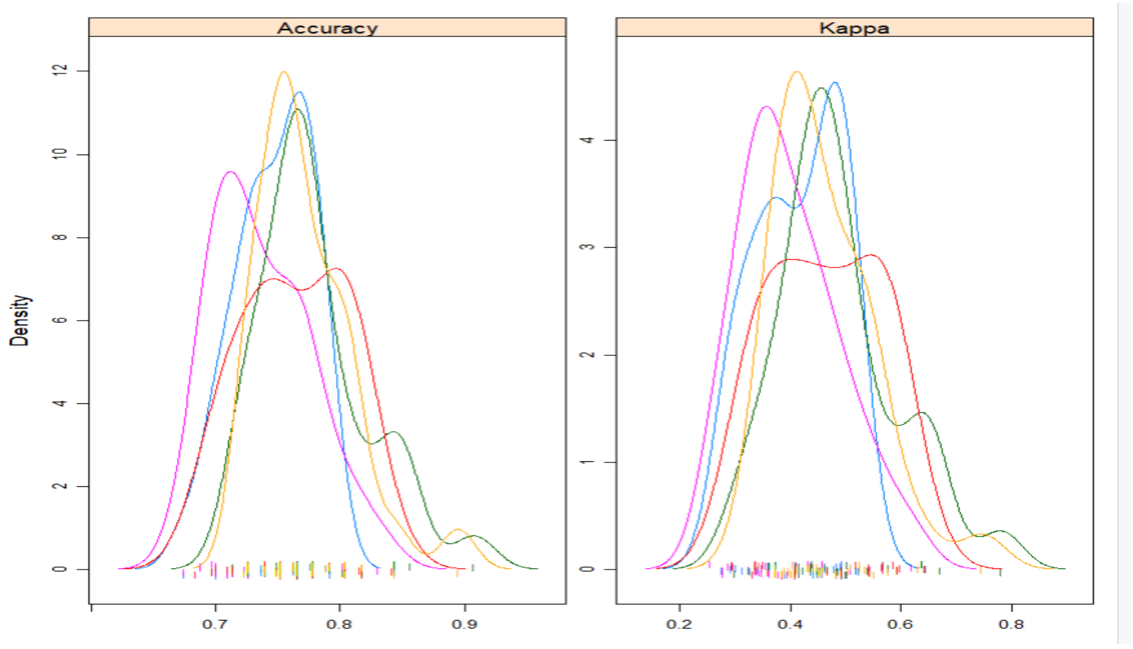


CODE:

# density plots of accuracy

```
scales <- list(x=list(relation="free"), y=list(relation="free"))
densityplot(results, scales=scales, pch = "|")
```

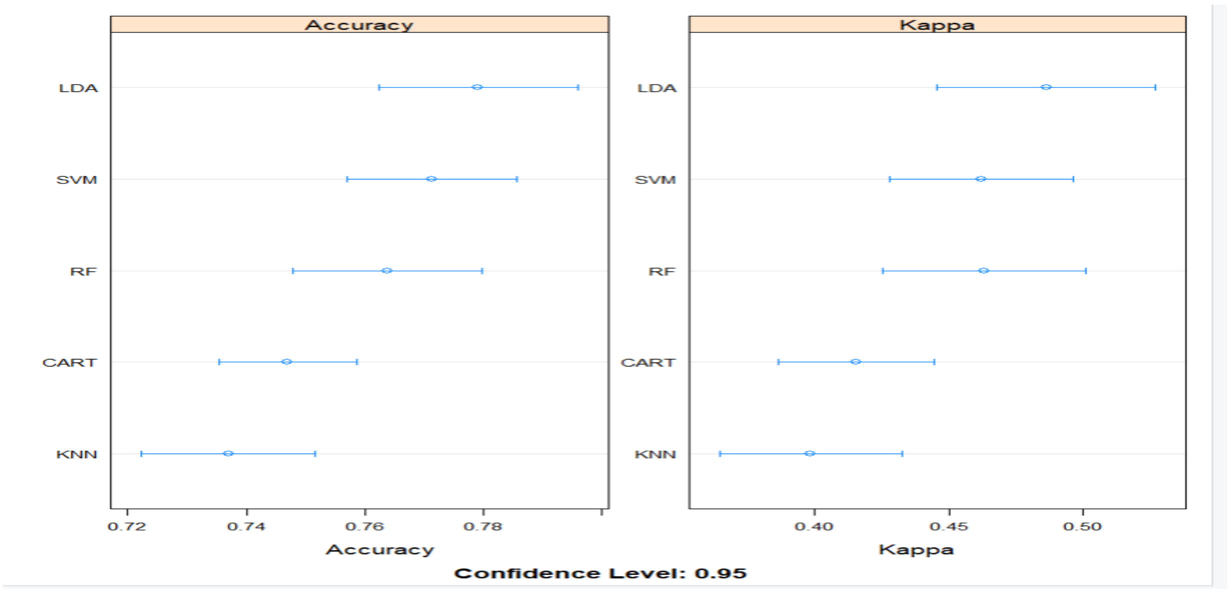
OUTPUT:



CODE:

```
# dot plots of accuracy
scales <- list(x=list(relation="free"), y=list(relation="free"))
dotplot(results, scales=scales)
```

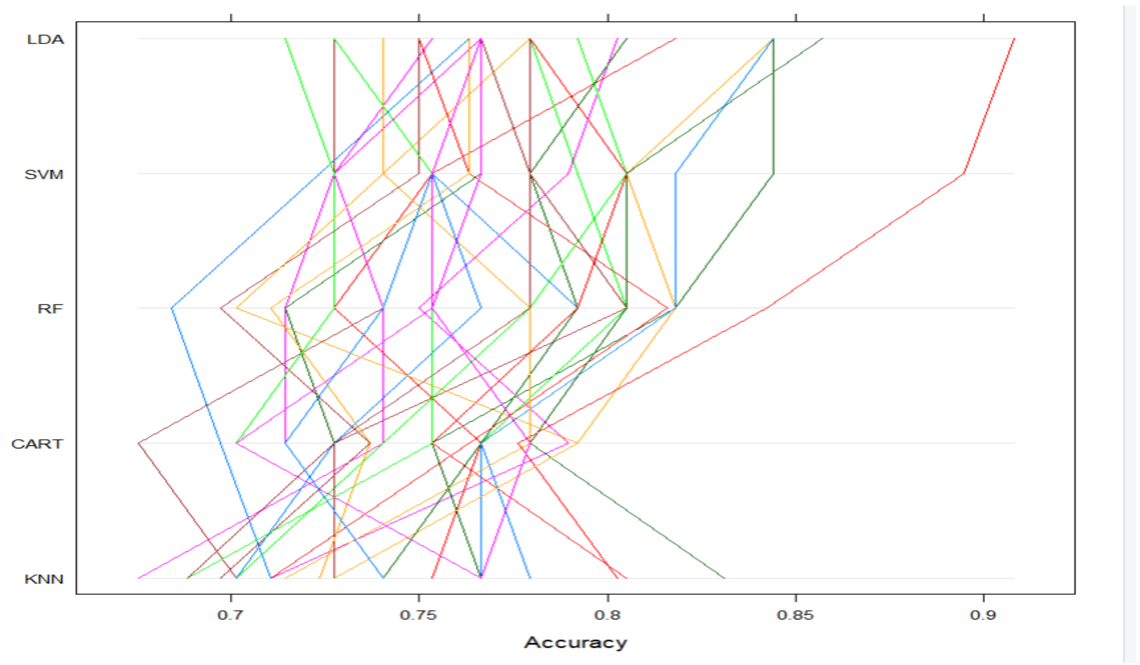
OUTPUT:



CODE:

```
parallelplot(results)   # parallel plots to compare models
```

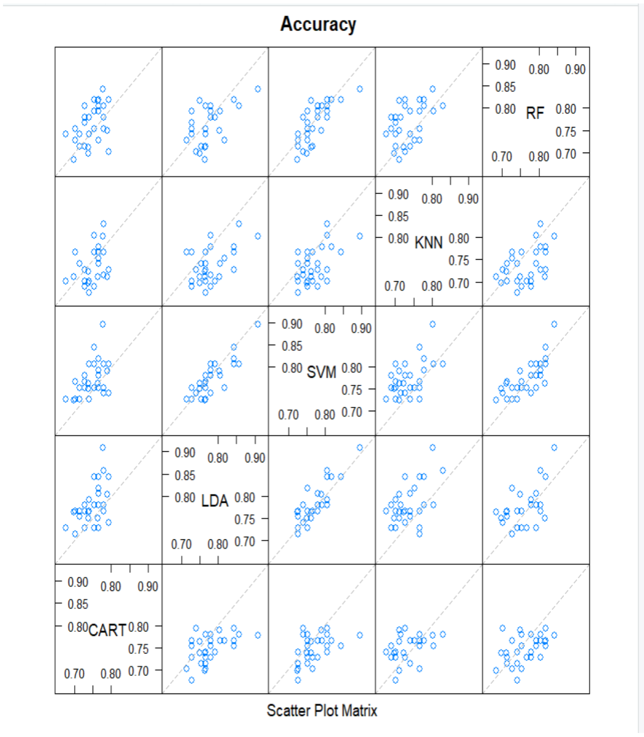
OUTPUT:

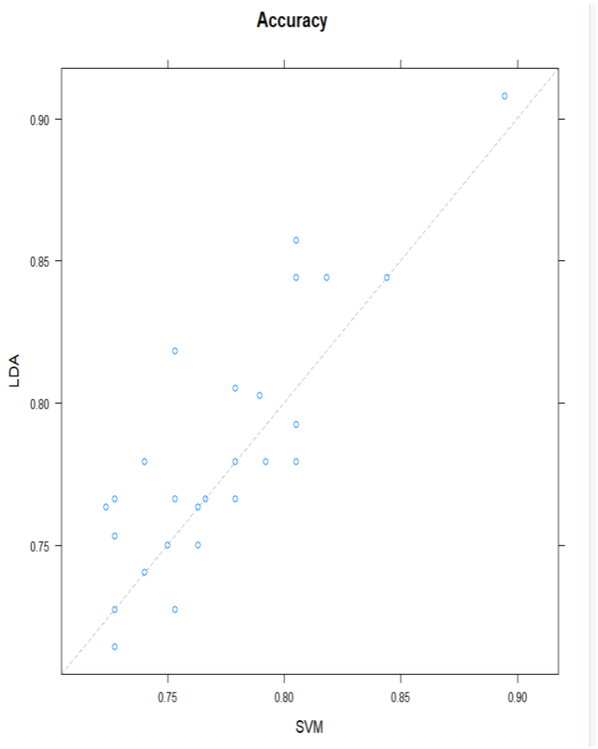


CODE:

```
splom(results)   # pair-wise scatterplots of predictions to compare models

xyplot(results, models=c("LDA", "SVM"))   # xyplot plots to compare models
```

OUTPUT:



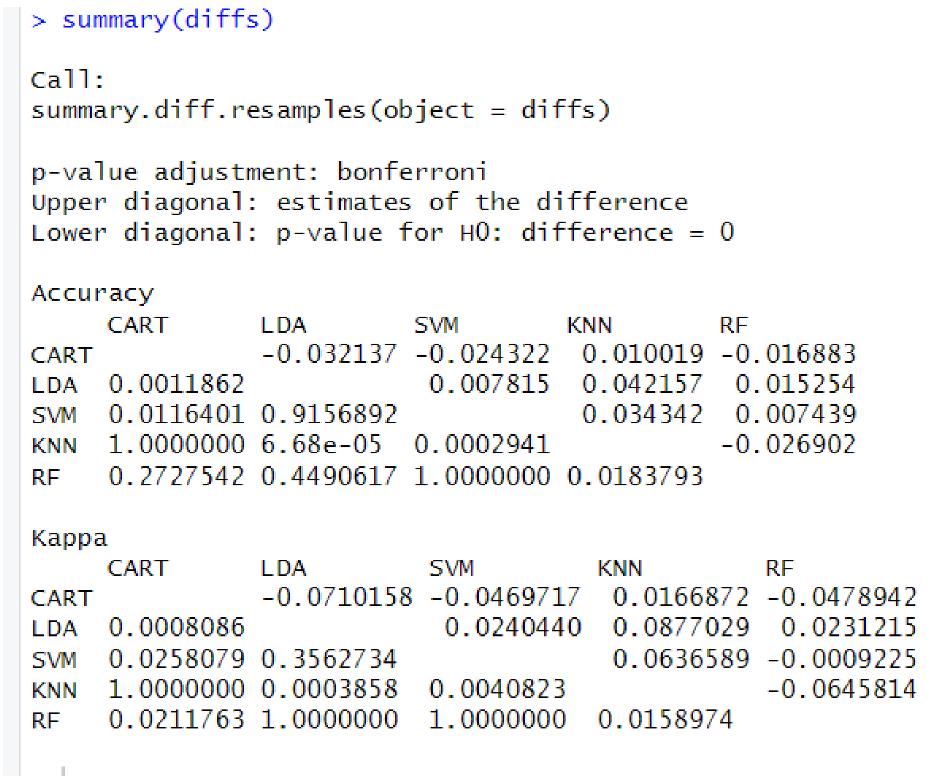


CODE:

diffs <- diff(results) **# difference in model predictions**

summary(diffs) **# summarize p-values for pair-wise comparisons**

OUTPUT:



```
> summary(diffs)

Call:
summary.diff.resamples(object = diffs)

p-value adjustment: bonferroni
Upper diagonal: estimates of the difference
Lower diagonal: p-value for H0: difference = 0

Accuracy
     CART      LDA       SVM       KNN       RF
CART           -0.032137 -0.024322  0.010019 -0.016883
LDA  0.0011862             0.007815  0.042157  0.015254
SVM  0.0116401 0.9156892            0.034342  0.007439
KNN  1.0000000 6.68e-05  0.0002941           -0.026902
RF   0.2727542 0.4490617 1.0000000 0.0183793

Kappa
     CART      LDA        SVM        KNN        RF
CART           -0.0710158 -0.0469717  0.0166872 -0.0478942
LDA  0.0008086              0.0240440  0.0877029  0.0231215
SVM  0.0258079 0.3562734              0.0636589 -0.0009225
KNN  1.0000000 0.0003858  0.0040823             -0.0645814
RF   0.0211763 1.0000000  1.0000000  0.0158974
```

EXPERIMENT NO:12

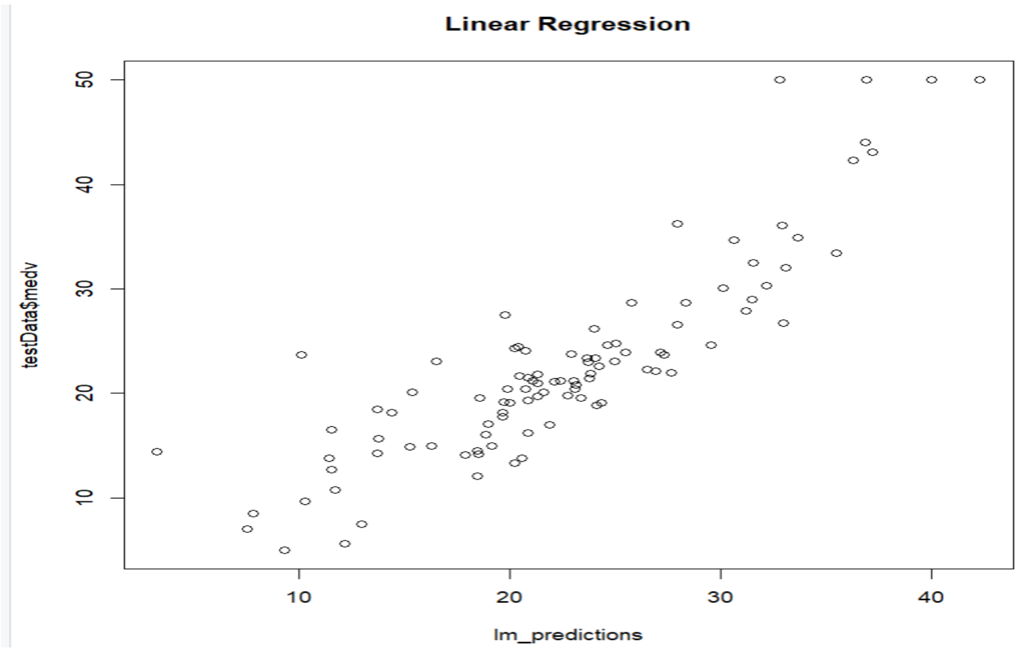
AIM: Create aR program to implement Regression Comparison on Boston Housing dataset and compare and visualize the results

DESCRIPTION: The “**Boston”** Housing dataset is a commonly used dataset for regression tasks in machine learning. It contains various features related to housing in Boston, such as the crime rate, average number of rooms per dwelling, and more. The target variable in this dataset is typically the median value of owner-occupied homes (in thousands of dollars).

CODE:

```
# Load necessary libraries

library(MASS)
library(caret)
library(e1071)
library(randomForest)
library(rpart)
library(rpart.plot)
data(Boston)   # Load BostonHousing dataset
set.seed(123)  # Set seed for reproducibility
# Split data into training and testing sets

trainIndex<- createDataPartition(Boston$medv, p = 0.8, list = FALSE)
trainData<- Boston[trainIndex, ]
testData<- Boston[-trainIndex, ]
# Linear Regression

lm_model<- lm(medv ~ ., data = trainData)
lm_predictions<- predict(lm_model, newdata = testData)
plot(lm_predictions, testData$medv, main = "Linear Regression")
```

OUTPUT:

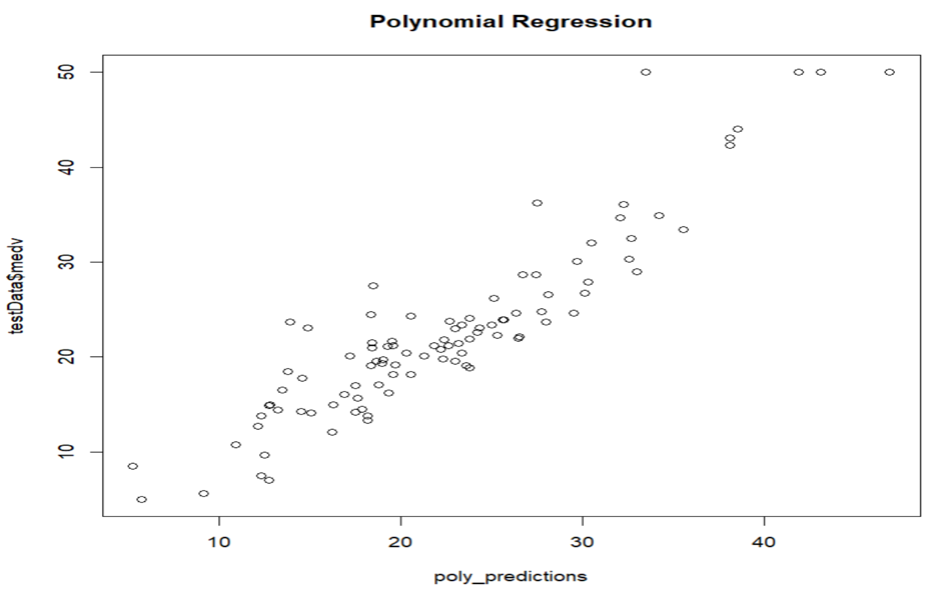


CODE:

# Polynomial Regression (excluding 'chas')

```
poly_model<- lm(medv ~ poly(crim, 2) + poly(zn, 2) + poly(indus, 2) + poly(nox, 2) + poly(rm, 2) + poly(age, 2) +
poly(dis, 2) + poly(rad, 2) + poly(tax, 2) + poly(ptratio, 2) + poly(black, 2) + poly(lstat, 2), data = trainData)
poly_predictions<- predict(poly_model, newdata = testData)
plot(poly_predictions, testData$medv, main = "Polynomial Regression")
```

OUTPUT:

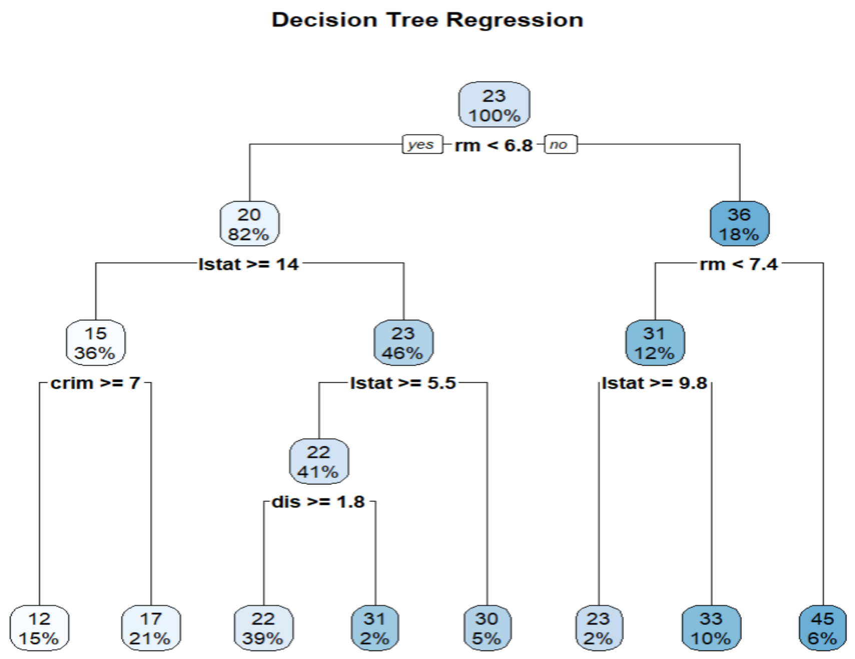


CODE:

# Decision Tree Regression

```
tree_model<- rpart(medv ~ ., data = trainData, method = "anova")
tree_predictions<- predict(tree_model, newdata = testData)
rpart.plot(tree_model, main = "Decision Tree Regression")
```

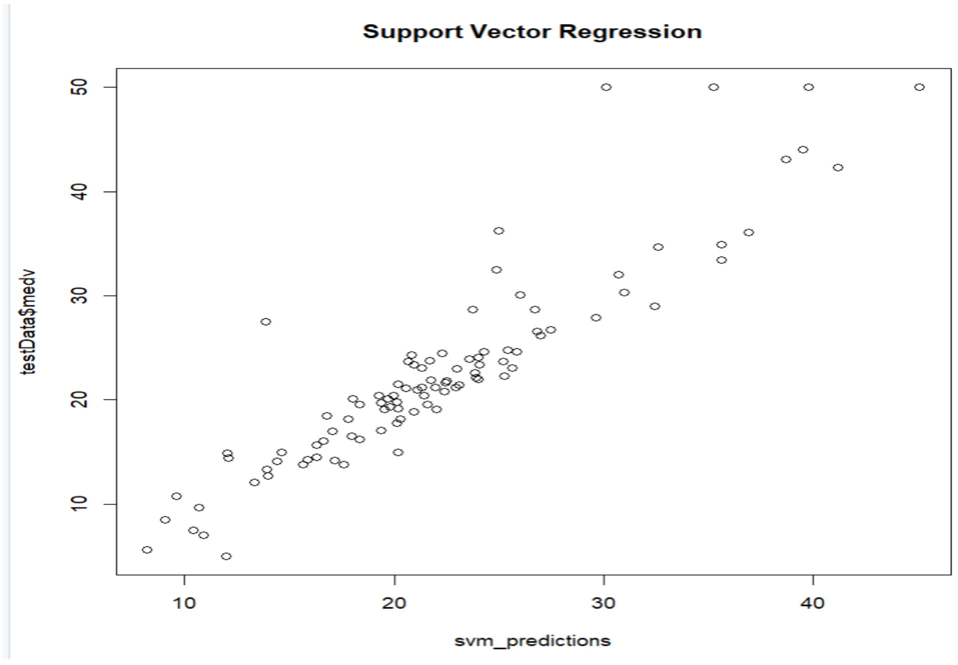
OUTPUT:



CODE:

```
# Support Vector Regression

svm_model<- svm(medv ~ ., data = trainData)

svm_predictions<- predict(svm_model, newdata = testData)

plot(svm_predictions, testData$medv, main = "Support Vector Regression")
```

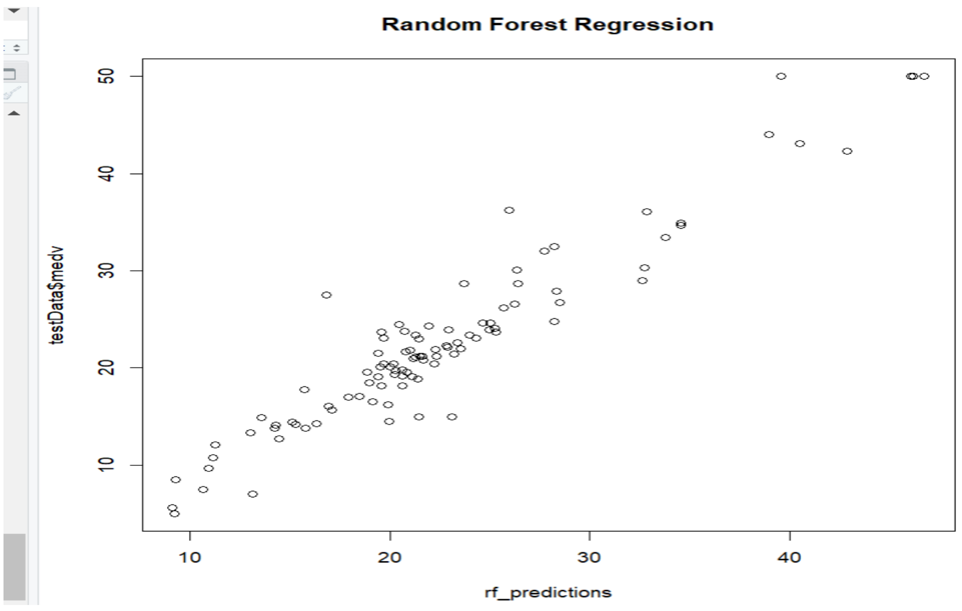
OUTPUT:



CODE:

```
# Random Forest Regression

rf_model<- randomForest(medv ~ ., data = trainData)
rf_predictions<- predict(rf_model, newdata = testData)
plot(rf_predictions, testData$medv, main = "Random Forest Regression")
```

OUTPUT:



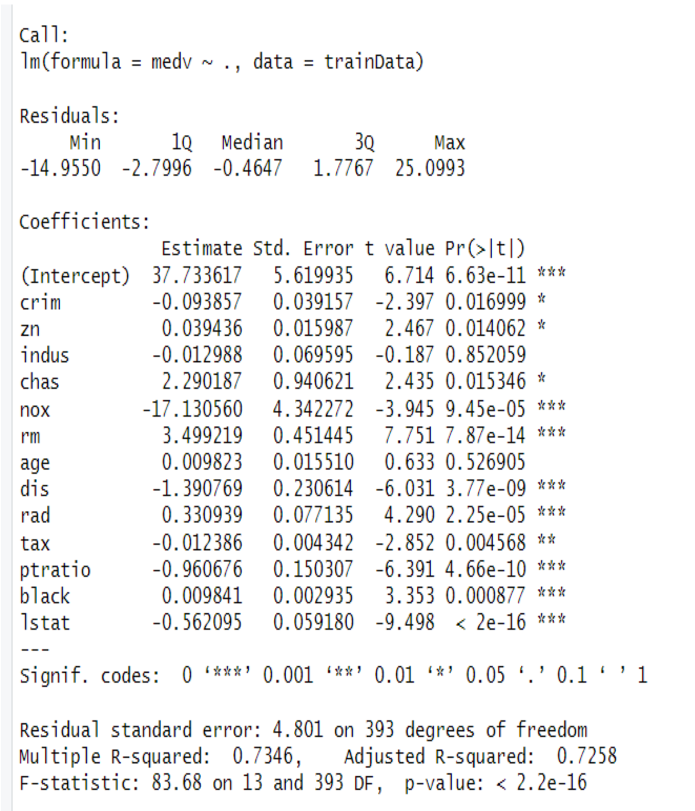
CODE:

# Model Summaries

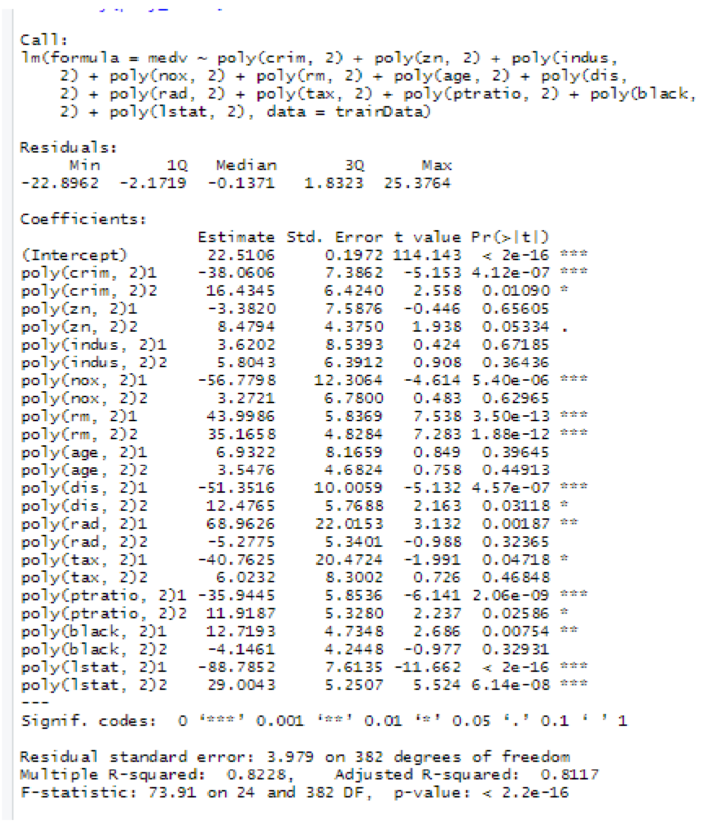
summary(lm_model)

summary(poly_model)

OUTPUT:



```
Call:
lm(formula = medv ~ ., data = trainData)

Residuals:
     Min       1Q   Median       3Q      Max 
-14.9550  -2.7996  -0.4647   1.7767  25.0993 

Coefficients:
              Estimate Std. Error t value Pr(>|t|)    
(Intercept)  37.733617   5.619935   6.714 6.63e-11 ***
crim         -0.093857   0.039157  -2.397 0.016999 *  
zn            0.039436   0.015987   2.467 0.014062 *  
indus        -0.012988   0.069595  -0.187 0.852059    
chas          2.290187   0.940621   2.435 0.015346 *  
nox         -17.130560   4.342272  -3.945 9.45e-05 ***
rm            3.499219   0.451445   7.751 7.87e-14 ***
age           0.009823   0.015510   0.633 0.526905    
dis          -1.390769   0.230614  -6.031 3.77e-09 ***
rad           0.330939   0.077135   4.290 2.25e-05 ***
tax          -0.012386   0.004342  -2.852 0.004568 ** 
ptratio      -0.960676   0.150307  -6.391 4.66e-10 ***
black         0.009841   0.002935   3.353 0.000877 ***
lstat        -0.562095   0.059180  -9.498  < 2e-16 ***
---
Signif. codes:  0 ‘***’ 0.001 ‘**’ 0.01 ‘*’ 0.05 ‘.’ 0.1 ‘ ’ 1

Residual standard error: 4.801 on 393 degrees of freedom
Multiple R-squared:  0.7346,	Adjusted R-squared:  0.7258 
F-statistic: 83.68 on 13 and 393 DF,  p-value: < 2.2e-16
```

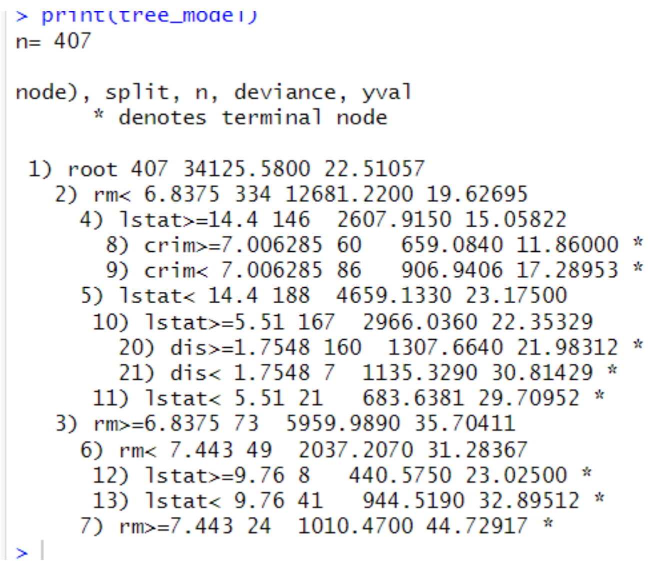


```
Call:
lm(formula = medv ~ poly(crim, 2) + poly(zn, 2) + poly(indus, 
    2) + poly(nox, 2) + poly(rm, 2) + poly(age, 2) + poly(dis, 
    2) + poly(rad, 2) + poly(tax, 2) + poly(ptratio, 2) + poly(black, 
    2) + poly(lstat, 2), data = trainData)

Residuals:
     Min       1Q   Median       3Q      Max 
-22.8962  -2.1719  -0.1371   1.8323  25.3764 

Coefficients:
                   Estimate Std. Error t value Pr(>|t|)    
(Intercept)         22.5106     0.1972 114.143  < 2e-16 ***
poly(crim, 2)1     -38.0606     7.3862  -5.153 4.12e-07 ***
poly(crim, 2)2      16.4345     6.4240   2.558  0.01090 *  
poly(zn, 2)1        -3.3820     7.5876  -0.446  0.65605    
poly(zn, 2)2         8.4794     4.3750   1.938  0.05334 .  
poly(indus, 2)1      3.6202     8.5393   0.424  0.67185    
poly(indus, 2)2      5.8043     6.3912   0.908  0.36436    
poly(nox, 2)1      -56.7798    12.3064  -4.614 5.40e-06 ***
poly(nox, 2)2        3.2721     6.7800   0.483  0.62965    
poly(rm, 2)1        43.9986     5.8369   7.538 3.50e-13 ***
poly(rm, 2)2        35.1658     4.8284   7.283 1.88e-12 ***
poly(age, 2)1        6.9322     8.1659   0.849  0.39645    
poly(age, 2)2        3.5476     4.6824   0.758  0.44913    
poly(dis, 2)1      -51.3516    10.0059  -5.132 4.57e-07 ***
poly(dis, 2)2       12.4765     5.7688   2.163  0.03118 *  
poly(rad, 2)1       68.9626    22.0153   3.132  0.00187 ** 
poly(rad, 2)2       -5.2775     5.3401  -0.988  0.32365    
poly(tax, 2)1      -40.7625    20.4724  -1.991  0.04718 *  
poly(tax, 2)2        6.0232     8.3002   0.726  0.46848    
poly(ptratio, 2)1 -35.9445     5.8536  -6.141 2.06e-09 ***
poly(ptratio, 2)2  11.9187     5.3280   2.237  0.02586 *  
poly(black, 2)1     12.7193     4.7348   2.686  0.00754 ** 
poly(black, 2)2     -4.1461     4.2448  -0.977  0.32931    
poly(lstat, 2)1    -88.7852     7.6135 -11.662  < 2e-16 ***
poly(lstat, 2)2     29.0043     5.2507   5.524 6.14e-08 ***
---
Signif. codes:  0 ‘***’ 0.001 ‘**’ 0.01 ‘*’ 0.05 ‘.’ 0.1 ‘ ’ 1

Residual standard error: 3.979 on 382 degrees of freedom
Multiple R-squared:  0.8228,	Adjusted R-squared:  0.8117 
F-statistic: 73.91 on 24 and 382 DF,  p-value: < 2.2e-16
```

CODE:

print(tree_model)

OUTPUT:



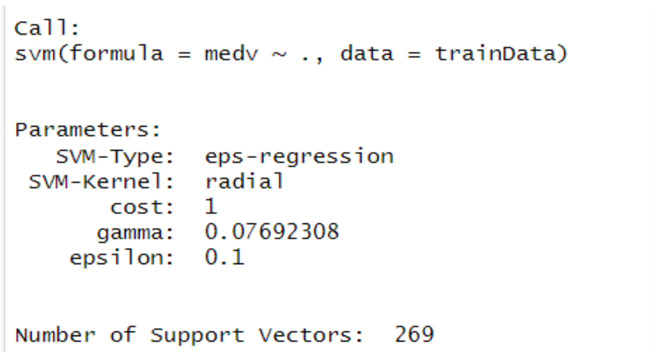
```
> print(tree_model)
n= 407 

node), split, n, deviance, yval
      * denotes terminal node

 1) root 407 34125.5800 22.51057  
   2) rm< 6.8375 334 12681.2200 19.62695  
     4) lstat>=14.4 146  2607.9150 15.05822  
       8) crim>=7.006285 60   659.0840 11.86000 *
       9) crim< 7.006285 86   906.9406 17.28953 *
     5) lstat< 14.4 188  4659.1330 23.17500  
      10) lstat>=5.51 167  2966.0360 22.35329  
        20) dis>=1.7548 160  1307.6640 21.98312 *
        21) dis< 1.7548 7  1135.3290 30.81429 *
      11) lstat< 5.51 21   683.6381 29.70952 *
   3) rm>=6.8375 73  5959.9890 35.70411  
     6) rm< 7.443 49  2037.2070 31.28367  
      12) lstat>=9.76 8   440.5750 23.02500 *
      13) lstat< 9.76 41   944.5190 32.89512 *
     7) rm>=7.443 24  1010.4700 44.72917 *
> |
```

CODE:

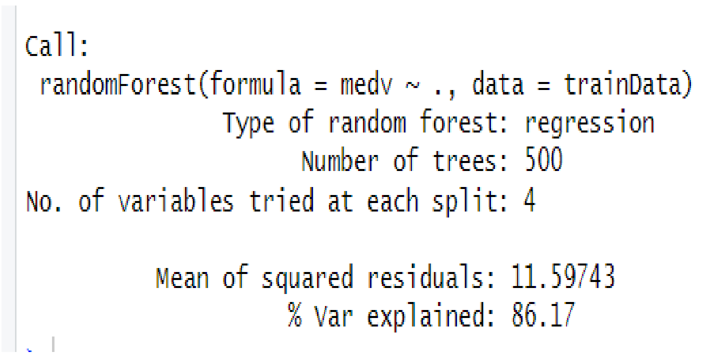
summary(svm_model)

print(rf_model)

OUTPUT:



```
Call:
svm(formula = medv ~ ., data = trainData)


Parameters:
   SVM-Type:  eps-regression
 SVM-Kernel:  radial
       cost:  1
      gamma:  0.07692308
    epsilon:  0.1


Number of Support Vectors:  269
```



```
Call:
 randomForest(formula = medv ~ ., data = trainData)
               Type of random forest: regression
                     Number of trees: 500
No. of variables tried at each split: 4

          Mean of squared residuals: 11.59743
                    % Var explained: 86.17
```

CODE:

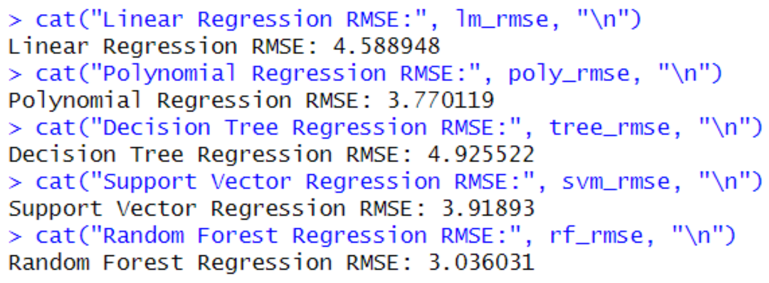
# Calculate RMSE for each model

lm_rmse<- rmse(testData$medv - lm_predictions)
poly_rmse<- rmse(testData$medv - poly_predictions)
tree_rmse<- rmse(testData$medv - tree_predictions)
svm_rmse<- rmse(testData$medv - svm_predictions)
rf_rmse<- rmse(testData$medv - rf_predictions)

# Print RMSEs for each model

cat("Linear Regression RMSE:",lm_rmse,"\n")
cat("Polynomial Regression RMSE:",ploy_rmse,"\n")
cat("Decision Tree Regression RMSE:",tree_rmse,"\n")
cat("Support Vector Regression RMSE:",svm_rmse,"\n")
cat("Random Forest Regression RMSE:",rf_rmse,"\n")

OUTPUT:



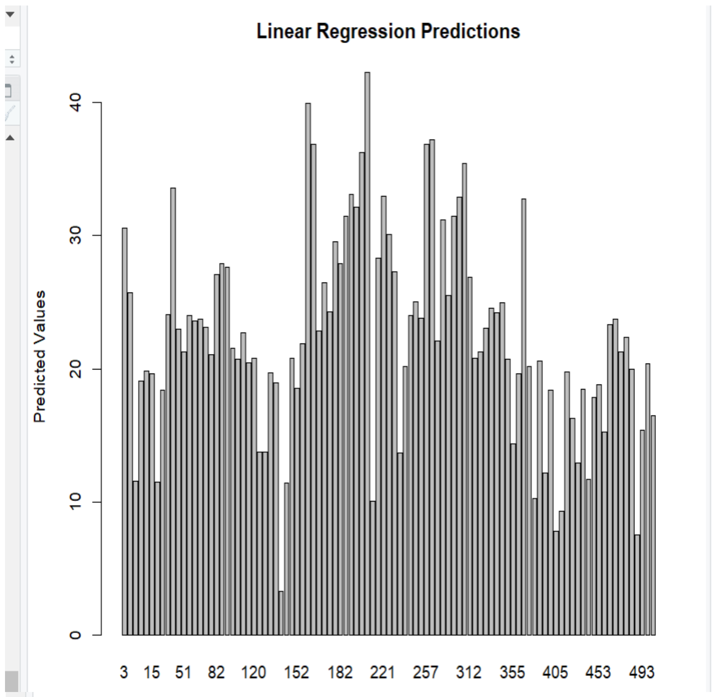
```
> cat("Linear Regression RMSE:", lm_rmse, "\n")
Linear Regression RMSE: 4.588948
> cat("Polynomial Regression RMSE:", poly_rmse, "\n")
Polynomial Regression RMSE: 3.770119
> cat("Decision Tree Regression RMSE:", tree_rmse, "\n")
Decision Tree Regression RMSE: 4.925522
> cat("Support Vector Regression RMSE:", svm_rmse, "\n")
Support Vector Regression RMSE: 3.91893
> cat("Random Forest Regression RMSE:", rf_rmse, "\n")
Random Forest Regression RMSE: 3.036031
```

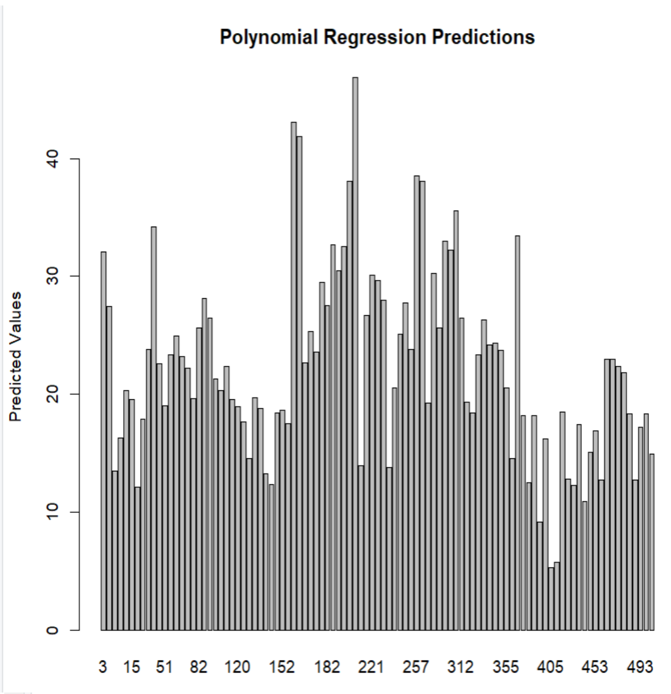
CODE:

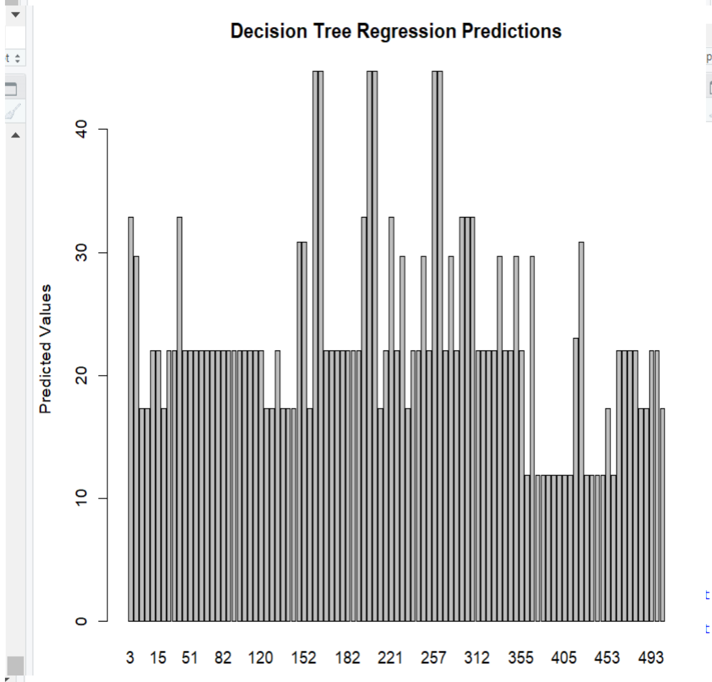
# Create bar graphs for each model's predictions

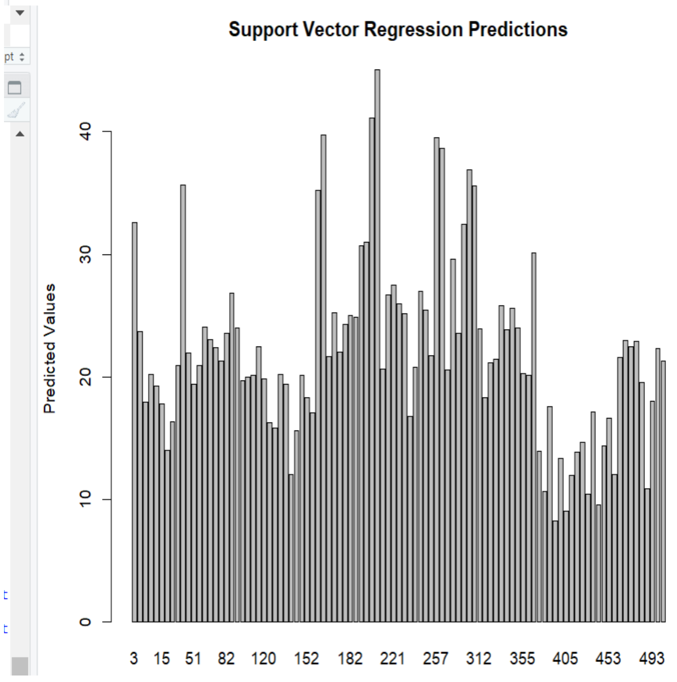
models <- c("Linear Regression", "Polynomial Regression", "Decision Tree Regression", "Support Vector Regression", "Random Forest Regression")
predictions <- list(lm_predictions, poly_predictions, tree_predictions, svm_predictions, rf_predictions)
 for (i in 1:length(models)) {
  par(mfrow=c(1,1))
barplot(predictions[[i]], main = paste(models[i], "Predictions"), ylab = "Predicted Values")}

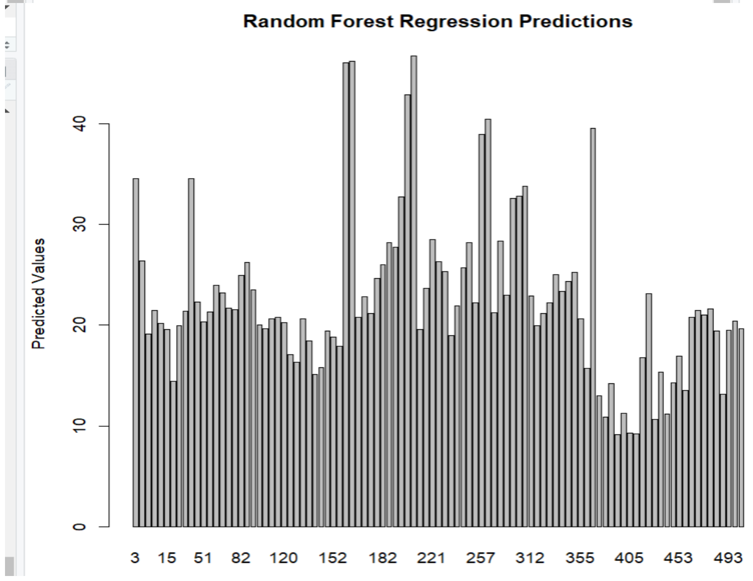
OUTPUT:


Linear Regression Predictions
Predicted Values
0
10
20
30
40
3 15 51 82 120 152 182 221 257 312 355 405 453 493


Polynomial Regression Predictions
Predicted Values
0
10
20
30
40
3 15 51 82 120 152 182 221 257 312 355 405 453 493


Decision Tree Regression Predictions
Predicted Values
0
10
20
30
40
3 15 51 82 120 152 182 221 257 312 355 405 453 493


Support Vector Regression Predictions
Predicted Values
0
10
20
30
40
3 15 51 82 120 152 182 221 257 312 355 405 453 493


Random Forest Regression Predictions
Predicted Values
0
10
20
30
40
3 15 51 82 120 152 182 221 257 312 355 405 453 493

EXPERIMENT NO:13

AIM: Create a R program to implement Clustering Comparison on College dataset and compare and visualize the results

DESCRIPTION: The "**College**" dataset is another commonly used dataset, typically for tasks like clustering and classification. This dataset contains information about various colleges in the United States. The dataset includes features related to the colleges, such as their names, types (public or private), number of students, acceptance rates, and more. One of the common uses of this dataset is for clustering colleges based on their characteristics.

CODE:

```
# Load required libraries

library(ISLR2)

library(cluster)

library(dbscan)

library(factoextra)

library(mclust)

library(ggplot2)

library(kohonen)

library(e1071)  # For FCM

library(reshape2)  # For data manipulation

data("College")  # Load the College dataset

# Select relevant features for clustering (excluding the college name)

college_data<- College[, -1]  # Exclude the first column (college name)

scaled_data<- scale(college_data)# Standardize the data

# Create a function to calculate silhouette scores

calculate_silhouette<- function(method, data, clusters) {

  if (!is.null(clusters)) {

silhouette_score<- mean(silhouette(clusters, dist(data)))

    return(silhouette_score)

  } else {

return(NA)

  }}

# Initialize a dataframe to store results

results <- data.frame(Method = character(0), Silhouette = numeric(0))
```

# Perform K-Means clustering

kmeans_result<- kmeans(scaled_data, centers = 3, nstart = 25)

kmeans_silhouette<- calculate_silhouette("K-Means", scaled_data, kmeans_result$cluster)

results <- rbind(results, data.frame(Method = "K-Means", Silhouette = kmeans_silhouette))

# Perform Hierarchical Clustering

hierarchical_result<- hclust(dist(scaled_data))

hierarchical_silhouette<- calculate_silhouette("Hierarchical", scaled_data, cutree(hierarchical_result, k = 3))

results <- rbind(results, data.frame(Method = "Hierarchical", Silhouette = hierarchical_silhouette))

# Perform DBSCAN clustering

dbscan_result<- dbscan(scaled_data, eps = 3, MinPts = 5)

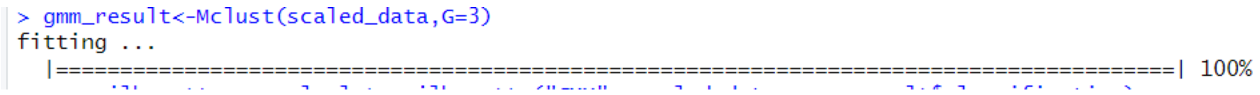
dbscan_silhouette<- calculate_silhouette("DBSCAN", scaled_data, dbscan_result$cluster)

results <- rbind(results, data.frame(Method = "DBSCAN", Silhouette = dbscan_silhouette))

# Perform Gaussian Mixture Model (GMM) clustering

gmm_result<-Mclust(scaled_data,G=3)

<u>OUTPUT:</u>



```
> gmm_result<-Mclust(scaled_data,G=3)
fitting ...
  |===================================================================================| 100%
```

gmm_silhouette<- calculate_silhouette("GMM", scaled_data, gmm_result$classification)

results <- rbind(results, data.frame(Method = "GMM", Silhouette = gmm_silhouette))

# Perform Self-Organizing Maps (SOM) clustering

som_result<- som(scale(scaled_data), grid = somgrid(3, 3, "hexagonal"))

som_silhouette<- calculate_silhouette("SOM", scale(scaled_data), som_result$unit.classif)

results <- rbind(results, data.frame(Method = "SOM", Silhouette = som_silhouette))

# Perform Fuzzy C-Means (FCM) clustering

fcm_result<- cmeans(scaled_data, centers = 3, m = 1.2)

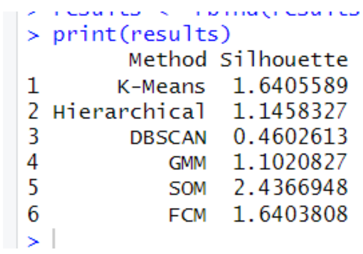
fcm_clusters<- fcm_result$cluster

fcm_silhouette<- calculate_silhouette("FCM", scaled_data, fcm_clusters)

results <- rbind(results, data.frame(Method = "FCM", Silhouette = fcm_silhouette))

Print(results)

OUTPUT:



```
> results <- rbind(results, data.f
> print(results)
       Method Silhouette
1     K-Means  1.6405589
2 Hierarchical  1.1458327
3      DBSCAN  0.4602613
4         GMM  1.1020827
5         SOM  2.4366948
6         FCM  1.6403808
> |
```

CODE:

# Find the index of the method with the highest silhouette score

best_method_index<- which.max(results$Silhouette)

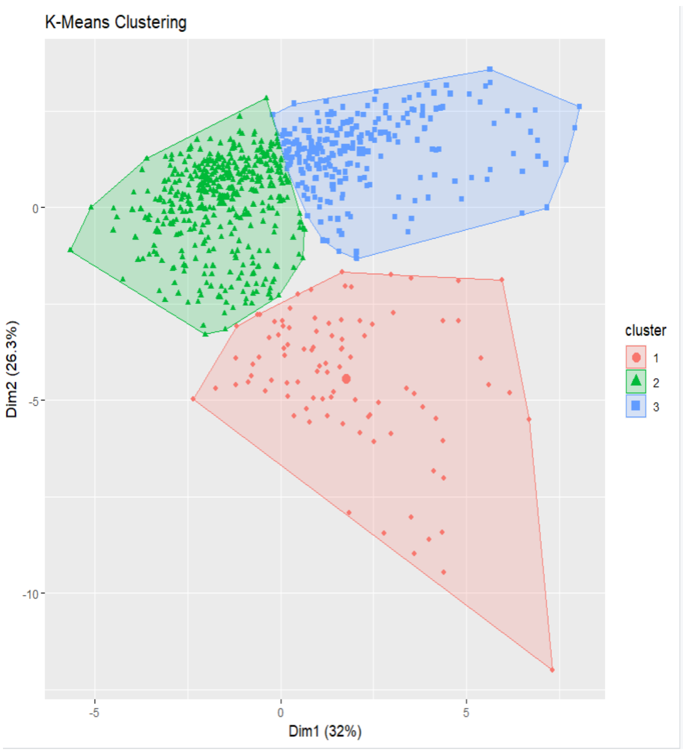
best_method<- results$Method[best_method_index]**# Retrieve the best method based on the index**

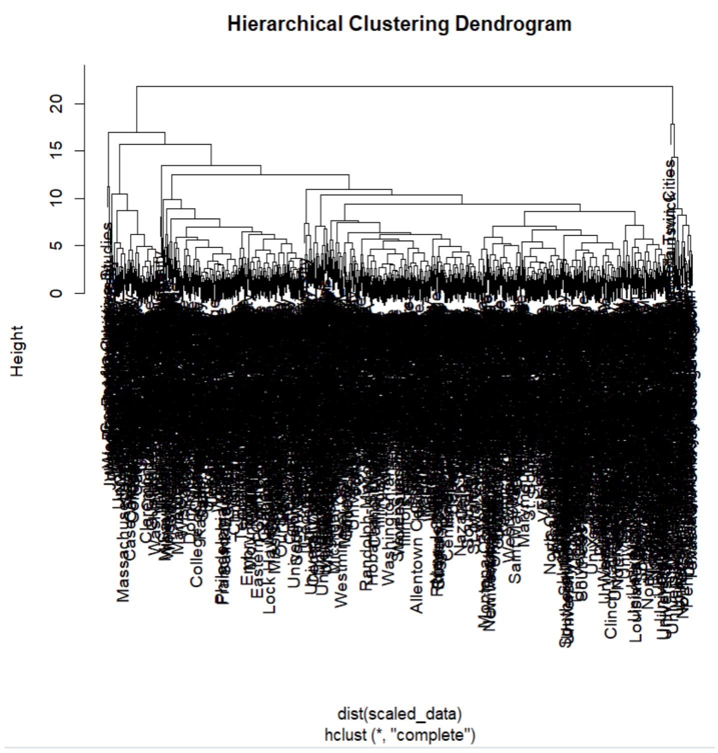
# Visualize K-Means clustering

fviz_cluster(kmeans_result, data = scaled_data, geom = "point", main = "K-Means Clustering")

plot(hierarchical_result, main = "Hierarchical Clustering Dendrogram")**# Visualize Hierarchical Clustering**

OUTPUT:





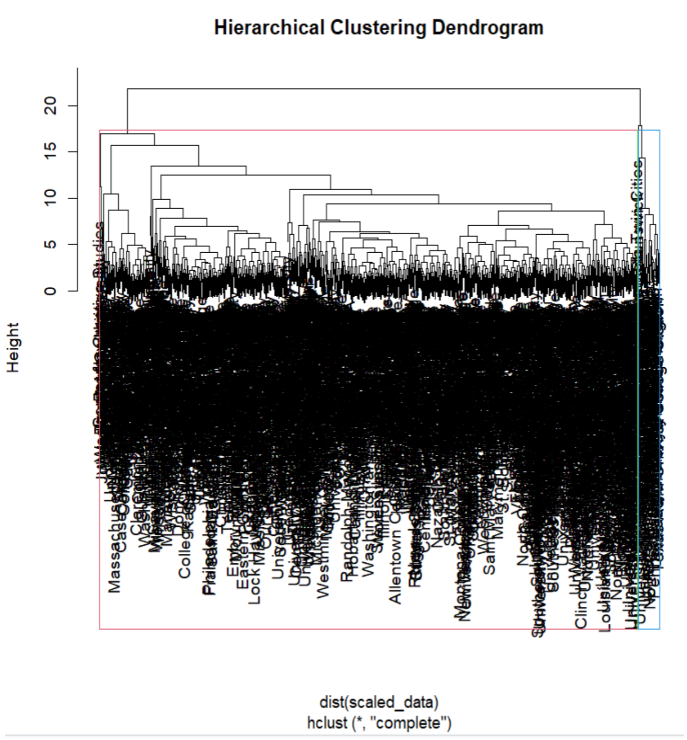
CODE:

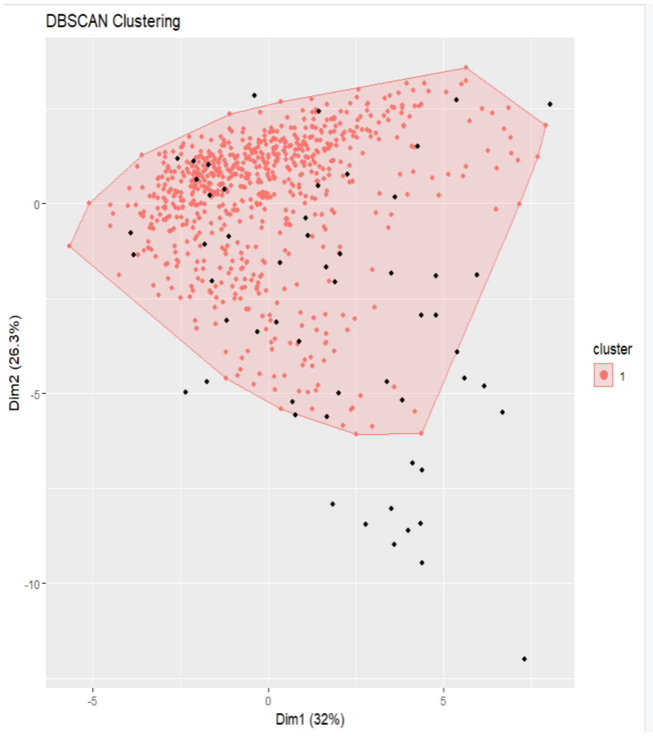
rect.hclust(hierarchical_result, k = 3, border = 2:4)

# Visualize DBSCAN clustering

fviz_cluster(dbscan_result, data = scaled_data, geom = "point", main = "DBSCAN Clustering")

OUTPUT:



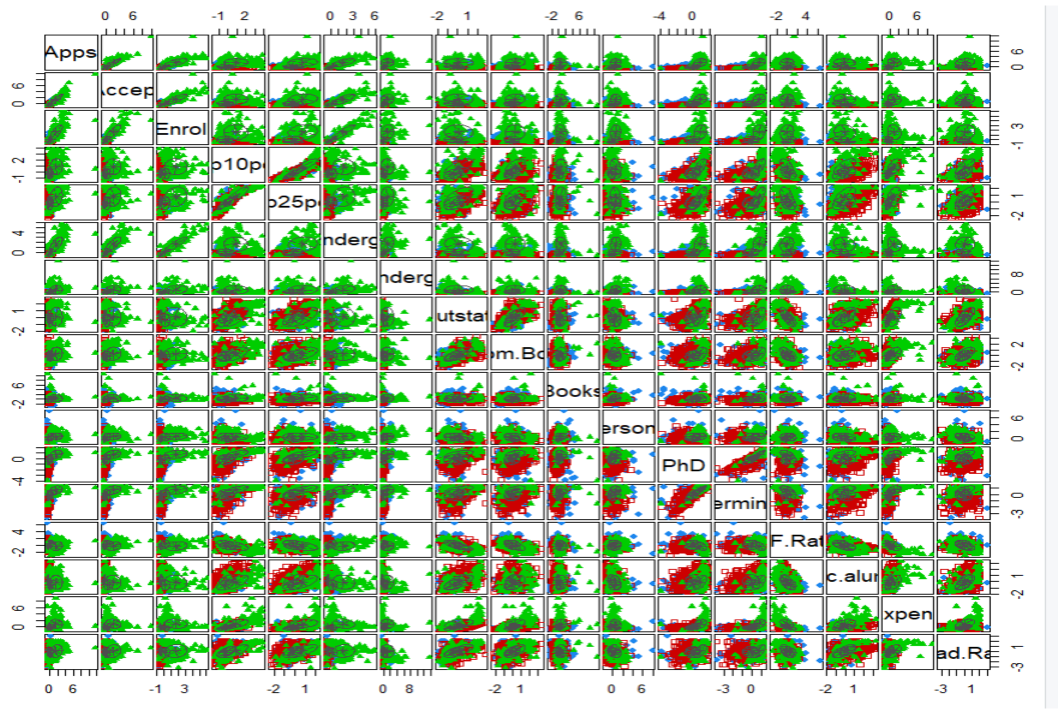


CODE:

# Visualize GMM clustering

```
plot(gmm_result, what = "classification", main = "GMM Clustering")
```

OUTPUT:

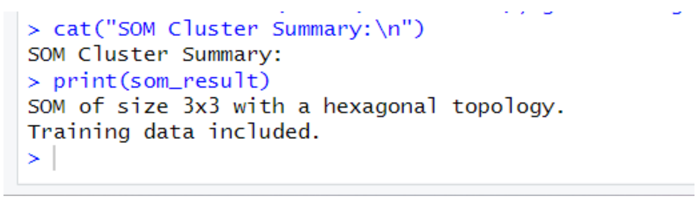


CODE:

# Perform Self-Organizing Maps (SOM) clustering

```
som_result<- som(scale(scaled_data), grid = somgrid(3, 3, "hexagonal")
cat(“SOM Cluster Summary:\n”)
print(som_result)
```

OUTPUT:



```
> cat("SOM Cluster Summary:\n")
SOM Cluster Summary:
> print(som_result)
SOM of size 3x3 with a hexagonal topology.
Training data included.
> |
```

CODE:

cluster_assignments<- som_result$unit.classif**# Get cluster assignments from the SOM result**

unique_clusters<- unique(cluster_assignments)**# Create a vector of unique cluster IDs**

cluster_colors<- rainbow(length(unique_clusters))**# Create a color palette for the clusters**

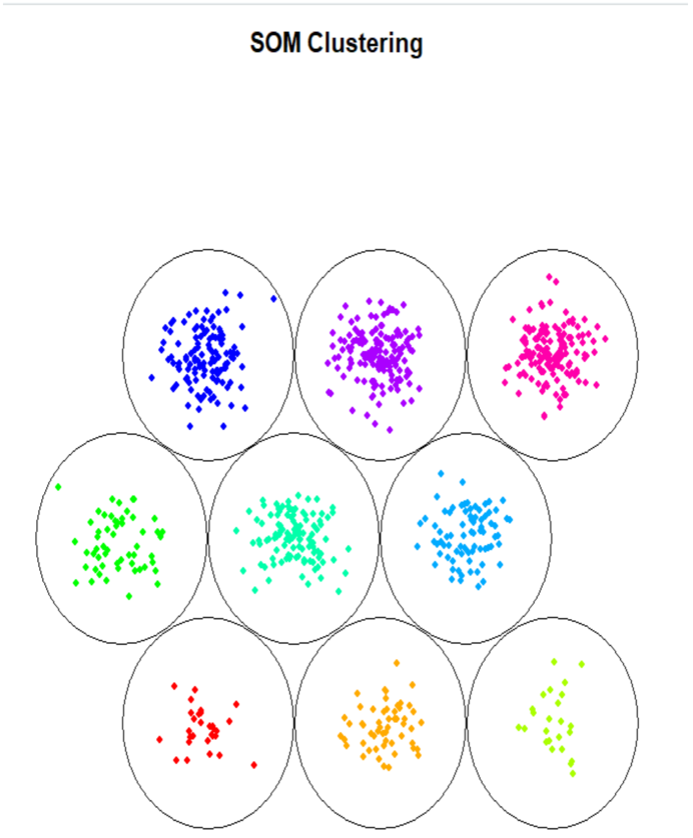
assigned_colors<- cluster_colors[cluster_assignments]**# Map cluster assignments to colors based on cluster IDs**

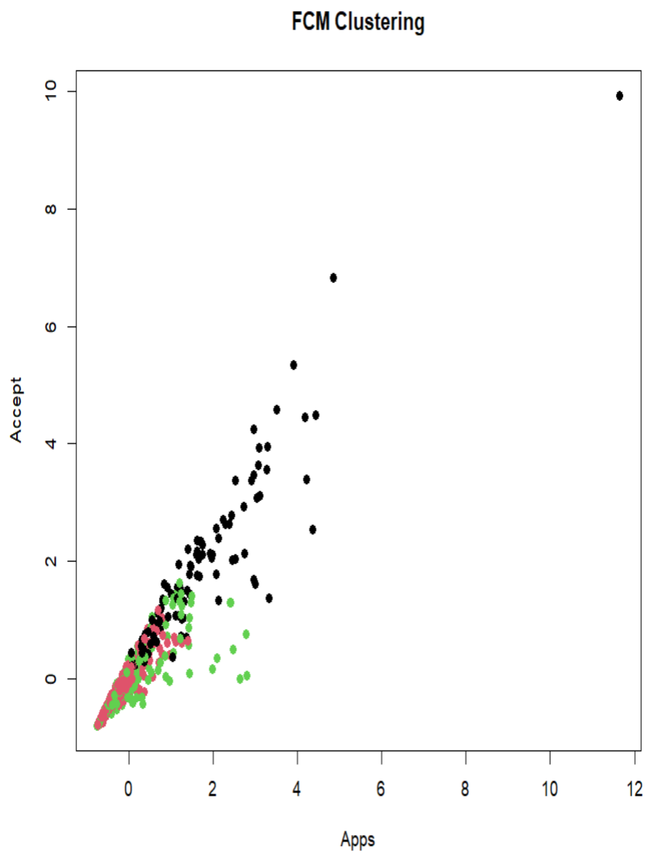
# Visualize SOM Clustering with assigned colors

plot(som_result, type = "mapping", pchs = 20, col = assigned_colors, main = "SOM Clustering")

plot(scaled_data, col = fcm_clusters, pch = 19, main = "FCM Clustering") **# Plot FCM clustering assignments**

OUTPUT:





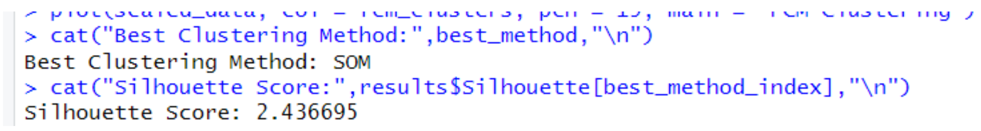
CODE:

# Print the best method and its silhouette score

Cat(“Best Clustering Method:”,best_method,”\n”)

Cat(“Silhouette score:”,results$Silhouette[best_method_index],”\n”)

OUTPUT:



```
> plot(scaled_data, col = fcm_clusters, pch = 19, main = "FCM Clustering")
> cat("Best Clustering Method:",best_method,"\n")
Best Clustering Method:  SOM
> cat("Silhouette Score:",results$Silhouette[best_method_index],"\n")
Silhouette Score:  2.436695
```

CODE:

# Create a bar plot of silhouette scores with different colors for each bar

ggplot(results, aes(x = Method, y = Silhouette, fill = Method)) +

geom_bar(stat = "identity") +

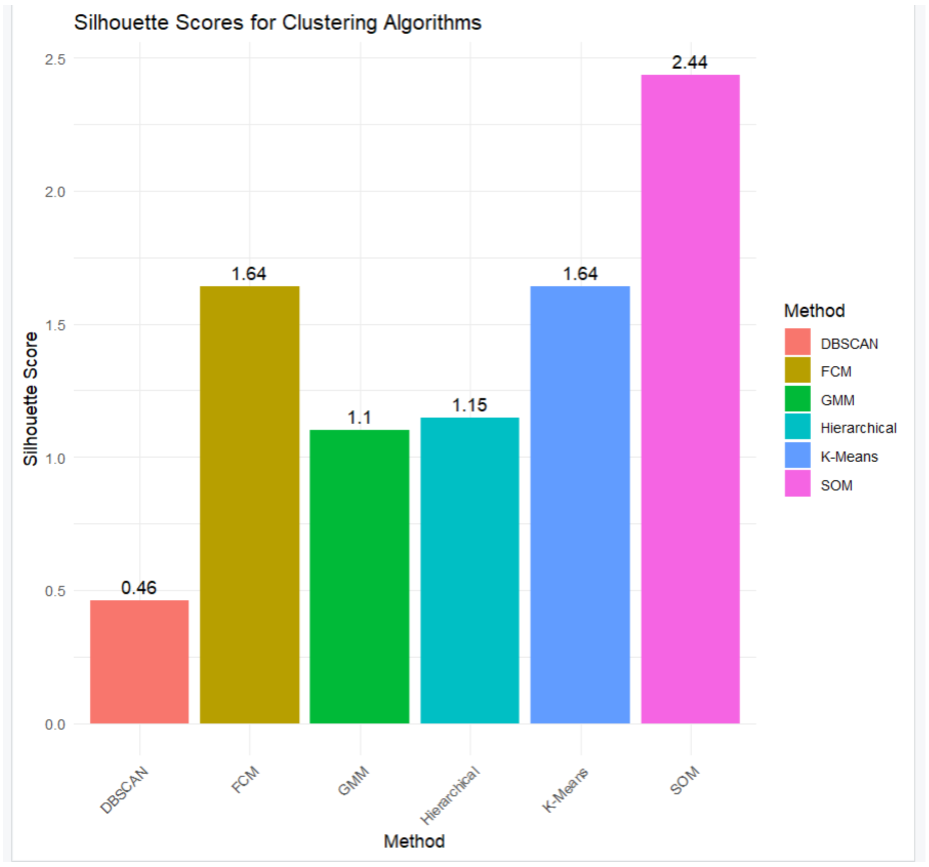
geom_text(aes(label = round(Silhouette, 2), vjust = -0.5), position = position_dodge(0.9)) + # Add labels on top of bars

labs(title = "Silhouette Scores for Clustering Algorithms", y = "Silhouette Score") +

theme_minimal() +

theme(axis.text.x = element_text(angle = 45, hjust = 1)) **# Rotate x-axis labels for better readability**

OUTPUT:


Silhouette Scores for Clustering Algorithms
Silhouette Score
DBSCAN
FCM
GMM
Hierarchical
K-Means
SOM
Method
Method
DBSCAN
FCM
GMM
Hierarchical
K-Means
SOM