EXPERIMENT NO: 8

AIM: To Implement Principal Component Analysis on the US Arrests data set.

DESCRIPTION:

Principal Component Analysis is an unsupervised learning algorithm that is used for the dimensionality reduction in machine learning . It is a statistical process that converts the observations of correlated features into a set of linearly uncorrelated features with the help of orthogonal transformation. These new transformed features are called the **Principal Components**. It is one of the popular tools that is used for exploratory data analysis and predictive modeling. It is a technique to draw strong patterns from the given dataset by reducing the variances.

PCA generally tries to find the lower-dimensional surface to project the high-dimensional data.

PCA works by considering the variance of each attribute because the high attribute shows the good split between the classes, and hence it reduces the dimensionality. Some real-world applications of PCA are ***image processing, movie recommendation system, optimizing the power allocation in various communication channels.*** It is a feature extraction technique, so it contains the important variables and drops the least important variable.

The PCA algorithm is based on some mathematical concepts such as:

- Variance and Covariance
- Eigenvalues and Eigen factors

Some common terms used in PCA algorithm:

- **Dimensionality:** It is the number of features or variables present in the given dataset. More easily, it is the number of columns present in the dataset.
- **Correlation:** It signifies that how strongly two variables are related to each other. Such as if one changes, the other variable also gets changed. The correlation value ranges from -1 to +1. Here, -1 occurs if variables are inversely proportional to each other, and +1 indicates that variables are directly proportional to each other.
- **Orthogonal:** It defines that variables are not correlated to each other, and hence the correlation between the pair of variables is zero.
- **Eigenvectors:** If there is a square matrix M, and a non-zero vector v is given. Then v will be eigenvector if Av is the scalar multiple of v.
- **Covariance Matrix:** A matrix containing the covariance between the pair of variables is called the Covariance Matrix.

Principal Components in PCA:

- As described above, the transformed new features or the output of PCA are the Principal Components. The number of these PCs are either equal to or less than the original features present in the dataset. Some properties of these principal components are given below:
- The principal component must be the linear combination of the original features.

- These components are orthogonal, i.e., the correlation between a pair of variables is zero.

- The importance of each component decreases when going to 1 to n, it means the 1 PC has the most importance, and n PC will have the least importance.

Steps for PCA algorithm:

1. **Getting the dataset**

   Firstly, we need to take the input dataset and divide it into two subparts X and Y, where X is the training set, and Y is the validation set.

2. **Representing data into a structure**

   Now we will represent our dataset into a structure. Such as we will represent the two-dimensional matrix of independent variable X. Here each row corresponds to the data items, and the column corresponds to the Features. The number of columns is the dimensions of the dataset.

3. **Standardizing the data**

   In this step, we will standardize our dataset. Such as in a particular column, the features with high variance are more important compared to the features with lower variance.

   If the importance of features is independent of the variance of the feature, then we will divide each data item in a column with the standard deviation of the column. Here we will name the matrix as Z.

4. **Calculating the Covariance of Z**

   To calculate the covariance of Z, we will take the matrix Z, and will transpose it. After transpose, we will multiply it by Z. The output matrix will be the Covariance matrix of Z.

5. **Calculating the Eigen Values and Eigen Vectors**

   Now we need to calculate the eigenvalues and eigenvectors for the resultant covariance matrix Z. Eigenvectors or the covariance matrix are the directions of the axes with high information. And the coefficients of these eigenvectors are defined as the eigenvalues.

6. **Sorting the Eigen Vectors**

   In this step, we will take all the eigenvalues and will sort them in decreasing order, which means from largest to smallest. And simultaneously sort the eigenvectors accordingly in matrix P of eigenvalues. The resultant matrix will be named as P*.

7. **Calculating the new features Or Principal Components**

   Here we will calculate the new features. To do this, we will multiply the P* matrix to the Z. In the resultant matrix Z*, each observation is the linear combination of original features. Each column of the Z* matrix is independent of each other.

8. **Remove less or unimportant features from the new dataset.**

   The new feature set has occurred, so we will decide here what to keep and what to remove. It means, we will only keep the relevant or important features in the new dataset, and unimportant features will be removed out. library(ggplot2) library(gridExtra) library(caret)

```
library(ISLR2)

#LOAD THE USARrrests dataset
data("USArrests")

#STANDARDISE THE DATA

scaled_data<-scale(USArrests)

#perform PCA

pca_result<-prcomp(scaled_data,center=TRUE,scale.=TRUE)

#summary the PCA result
summary(pca_result)

#variable explained by each principle component

variance_explained <- pca_result$sdev^2 / sum(pca_result$sdev^2) * 100

#print the variance explained by each pc cat("variance
explained by each principal component:\n")
print(variance_explained)
```

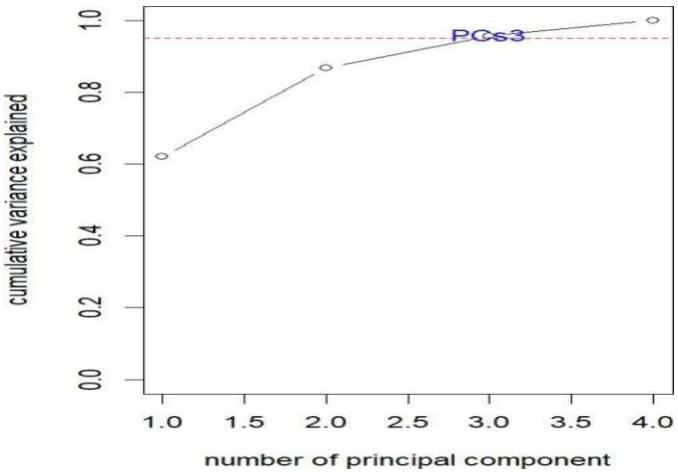
OUTPUT:

```
Importance of components:
                          PC1    PC2     PC3     PC4
Standard deviation     1.5749 0.9949 0.59713 0.41645
Proportion of Variance 0.6201 0.2474 0.08914 0.04336
Cumulative Proportion  0.6201 0.8675 0.95664 1.00000
> #variable explained by each principle component
> variance_explained <- pca_result$sdev^2 / sum(pca_result$sdev^2) * 100
> #print the variance explained by each pc
> cat("variance explained by each principal component:\n")
variance explained by each principal component:
> print(variance_explained)
[1] 62.006039 24.744129  8.914080  4.335752
```

```
#scree plot to visualize the variance explained by each pc
plot(variance_explained,type = "b",xlab="principal component",ylab = "variance explained(%)")

#calculate the cumulative proportion of variance explained
variance_explained <- pca_result$sdev^2 / sum(pca_result$sdev^2)
cumulative_variance_explained <- cumsum(variance_explained)
plot(cumulative_variance_explained, type = "b",
    xlab = "number of principal component",
     ylab = "cumulative variance explained",ylim = c(0,1))
```

```
#add a horizontal line at 0.95 to visualize the 95% threshold
abline(h=0.95,col = "red",lty=2)


#find the number of pcs needed to explain atleast 95% of the variance
num_pcs_to_reach_95_percent <- which(cumulative_variance_explained
>=0.95)[1] text(num_pcs_to_reach_95_percent,0.96, labels =
paste0("PCs",num_pcs_to_reach_95_percent), col = "blue")
```

OUTPUT:



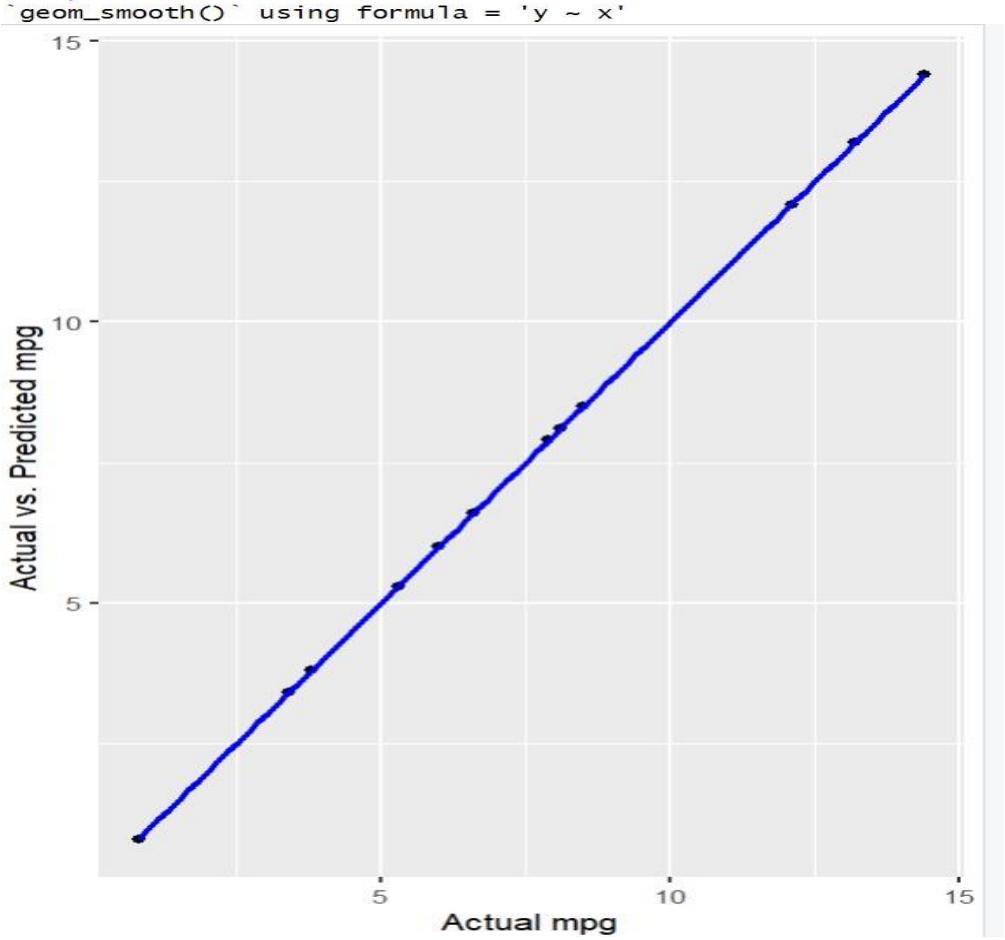
CODE:

```
#define the desired threshold (e.g,90%)
desired_threshold <- 0.90

#find the number of PCs needed to reach or exceed the desired threshold
num_pcs_to_reach_threshold <- which(cumulative_variance_explained >= desired_threshold)[1]

#extract the top "n" principal components
top_n_pcs <- pca_result$x[,1:num_pcs_to_reach_threshold]

#print the result you can now use top_n_pcs
cat("Number if PCs to convert at least",(desired_threshold * 100),
   "% variance:",num_pcs_to_reach_threshold,"\n")

#Retain the first 4 principal components
top_4_pcs <- pca_result$x[, 1:4]

#create a data frame with the top 4 PCs and the mpg variable
data_with_pcs <- data.frame(top_4_pcs,mpg = USArrests$Murder)


#split the data into a training set (75%) nd a testing set(25%)
set.seed(123)
```

```
train_index <- createDataPartition(data_with_pcs$mpg, p =0.75, list =
FALSE) train_data <- data_with_pcs[train_index, ] test_data <-
data_with_pcs[-train_index, ] model <- lm(mpg ~ . , data = train_data)
predictions <- predict(model, newdata = test_data) rmse <-
sqrt(mean((test_data$mpg - predictions)^2)) cat("Root Mean Squared Error
(RMSE):", rmse , "\n") result_data <- data.frame(Actual = test_data$mpg,

                    Predicted = predictions)

ggplot(result_data, aes(x = Actual, y = Predicted)) +

geom_point() +

  geom_smooth(method = "lm", se = FALSE, color = "blue") +
  labs(

    x = "Actual mpg",
    y = "Actual vs. Predicted mpg"
  )

residuals <- test_data$mpg - predictions screen_plot <-
ggplot(NULL,    aes(x    =    1:ncol(top_4_pcs),    y    =
cumulative_variance_explained)) +
  geom_bar(stat = "identity", fill = "blue") +
  labs(

    x = "Principal Component", y =
    "Cumulative Variance Explained",

    title = "Scree Plot (PCA)"

  )

residuals_plot <- ggplot(NULL, aes(x = residuals)) +

  geom_histogram(binwidth = 1, fill = "green", color ="black") +
  labs(

    x = "Residuals", y = "Frequemcy",
    title = "Residual plot
    (Regression)"

  )

actual_vs_predicted_plot <- ggplot(result_data, aes(x= Actual,
                                    y = Predicted)) + geom_point() +

geom_smooth(method = "lm", se = FALSE, color = "blue") + labs(

    x = "Actual mpg", y = "Predicted
    mpg", title = "Actual vs.
    Predicted mpg"

  )
```

OUTPUT:

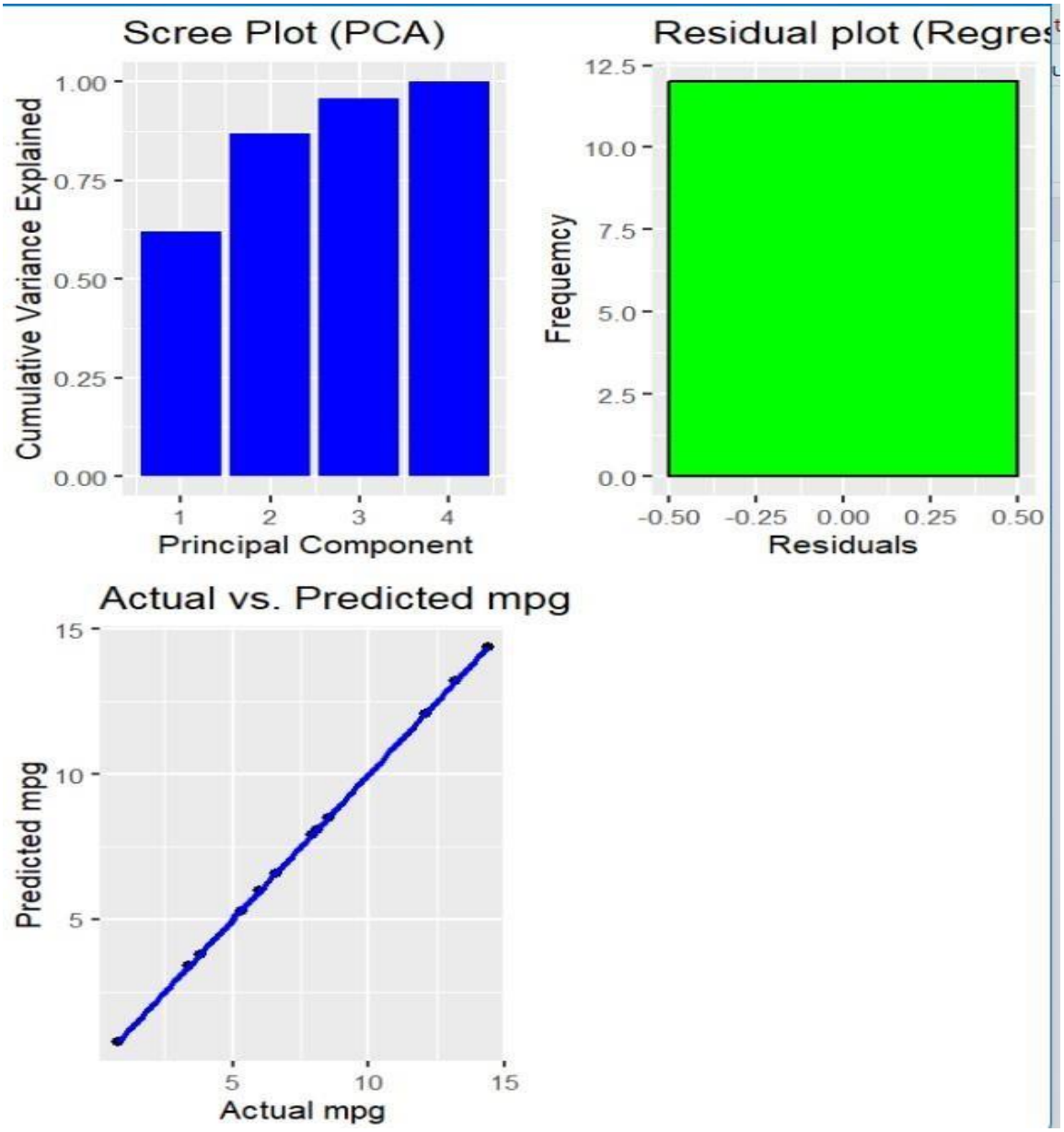
```
> #create a data frame with the top 4 PCs and the mpg variable
> data_with_pcs <- data.frame(top_4_pcs,mpg = USArrests$Murder)
>
> #split the data into a training set (75%) nd a testing set(25%)
> set.seed(123)
> train_index <- createDataPartition(data_with_pcs$mpg, p =0.75,
+                                    list = FALSE)
> train_data <- data_with_pcs[train_index, ]
> test_data <- data_with_pcs[-train_index, ]
> model <- lm(mpg ~ . , data = train_data)
> predictions <- predict(model, newdata = test_data)
> rmse <- sqrt(mean((test_data$mpg - predictions)^2))
> cat("Root Mean Squared Error (RMSE):", rmse , "\n")
Root Mean Squared Error (RMSE): 2.187121e-15
> result_data <- data.frame(Actual = test_data$mpg,
+                           Predicted = predictions)
> ggplot(result_data, aes(x = Actual, y = Predicted)) +
+   geom_point() +
+   geom_smooth(method = "lm", se = FALSE, color = "blue") +
+   labs(
+     x = "Actual mpg",
+     y = "Actual vs. Predicted mpg"
+   )
`geom_smooth()` using formula = 'y ~ x'
```



CODE:

```
summary_plot <- grid.arrange(screen_plot, residuals_plot,
                             actual_vs_predicted_plot, ncol = 2)

print(summary_plot)
```

OUTPUT:


Scree Plot (PCA)
Cumulative Variance Explained
1.00
0.75
0.50
0.25
0.00
1
2
3
4
Principal Component
Residual plot (Regres
Frequemcy
12.5
10.0
7.5
5.0
2.5
0.0
-0.50
-0.25
0.00
0.25
0.50
Residuals
Actual vs. Predicted mpg
Predicted mpg
15
10
5
5
10
15
Actual mpg