> wine

V1 V2 V3 V4 V5 V6 V7 V8 V9 V10 V11 V12 V13 V14

1 1 14.23 1.71 2.43 15.6 127 2.80 3.06 0.28 2.29 5.64 1.040 3.92 1065

2 1 13.20 1.78 2.14 11.2 100 2.65 2.76 0.26 1.28 4.38 1.050 3.40 1050

3 1 13.16 2.36 2.67 18.6 101 2.80 3.24 0.30 2.81 5.68 1.030 3.17 1185

4 1 14.37 1.95 2.50 16.8 113 3.85 3.49 0.24 2.18 7.80 0.860 3.45 1480

5 1 13.24 2.59 2.87 21.0 118 2.80 2.69 0.39 1.82 4.32 1.040 2.93 735

6 1 14.20 1.76 2.45 15.2 112 3.27 3.39 0.34 1.97 6.75 1.050 2.85 1450

7 1 14.39 1.87 2.45 14.6 96 2.50 2.52 0.30 1.98 5.25 1.020 3.58 1290

8 1 14.06 2.15 2.61 17.6 121 2.60 2.51 0.31 1.25 5.05 1.060 3.58 1295

9 1 14.83 1.64 2.17 14.0 97 2.80 2.98 0.29 1.98 5.20 1.080 2.85 1045

10 1 13.86 1.35 2.27 16.0 98 2.98 3.15 0.22 1.85 7.22 1.010 3.55 1045

11 1 14.10 2.16 2.30 18.0 105 2.95 3.32 0.22 2.38 5.75 1.250 3.17 1510

12 1 14.12 1.48 2.32 16.8 95 2.20 2.43 0.26 1.57 5.00 1.170 2.82 1280

13 1 13.75 1.73 2.41 16.0 89 2.60 2.76 0.29 1.81 5.60 1.150 2.90 1320

14 1 14.75 1.73 2.39 11.4 91 3.10 3.69 0.43 2.81 5.40 1.250 2.73 1150

15 1 14.38 1.87 2.38 12.0 102 3.30 3.64 0.29 2.96 7.50 1.200 3.00 1547

16 1 13.63 1.81 2.70 17.2 112 2.85 2.91 0.30 1.46 7.30 1.280 2.88 1310

17 1 14.30 1.92 2.72 20.0 120 2.80 3.14 0.33 1.97 6.20 1.070 2.65 1280

18 1 13.83 1.57 2.62 20.0 115 2.95 3.40 0.40 1.72 6.60 1.130 2.57 1130

19 1 14.19 1.59 2.48 16.5 108 3.30 3.93 0.32 1.86 8.70 1.230 2.82 1680

20 1 13.64 3.10 2.56 15.2 116 2.70 3.03 0.17 1.66 5.10 0.960 3.36 845

21 1 14.06 1.63 2.28 16.0 126 3.00 3.17 0.24 2.10 5.65 1.090 3.71 780

22 1 12.93 3.80 2.65 18.6 102 2.41 2.41 0.25 1.98 4.50 1.030 3.52 770

23 1 13.71 1.86 2.36 16.6 101 2.61 2.88 0.27 1.69 3.80 1.110 4.00 1035

24 1 12.85 1.60 2.52 17.8 95 2.48 2.37 0.26 1.46 3.93 1.090 3.63 1015

25 1 13.50 1.81 2.61 20.0 96 2.53 2.61 0.28 1.66 3.52 1.120 3.82 845

26 1 13.05 2.05 3.22 25.0 124 2.63 2.68 0.47 1.92 3.58 1.130 3.20 830

27 1 13.39 1.77 2.62 16.1 93 2.85 2.94 0.34 1.45 4.80 0.920 3.22 1195

28 1 13.30 1.72 2.14 17.0 94 2.40 2.19 0.27 1.35 3.95 1.020 2.77 1285

29 1 13.87 1.90 2.80 19.4 107 2.95 2.97 0.37 1.76 4.50 1.250 3.40 915

30 1 14.02 1.68 2.21 16.0 96 2.65 2.33 0.26 1.98 4.70 1.040 3.59 1035

31 1 13.73 1.50 2.70 22.5 101 3.00 3.25 0.29 2.38 5.70 1.190 2.71 1285

32 1 13.58 1.66 2.36 19.1 106 2.86 3.19 0.22 1.95 6.90 1.090 2.88 1515

33 1 13.68 1.83 2.36 17.2 104 2.42 2.69 0.42 1.97 3.84 1.230 2.87 990

34 1 13.76 1.53 2.70 19.5 132 2.95 2.74 0.50 1.35 5.40 1.250 3.00 1235

35 1 13.51 1.80 2.65 19.0 110 2.35 2.53 0.29 1.54 4.20 1.100 2.87 1095

36 1 13.48 1.81 2.41 20.5 100 2.70 2.98 0.26 1.86 5.10 1.040 3.47 920

37 1 13.28 1.64 2.84 15.5 110 2.60 2.68 0.34 1.36 4.60 1.090 2.78 880

38 1 13.05 1.65 2.55 18.0 98 2.45 2.43 0.29 1.44 4.25 1.120 2.51 1105

39 1 13.07 1.50 2.10 15.5 98 2.40 2.64 0.28 1.37 3.70 1.180 2.69 1020

40 1 14.22 3.99 2.51 13.2 128 3.00 3.04 0.20 2.08 5.10 0.890 3.53 760

41 1 13.56 1.71 2.31 16.2 117 3.15 3.29 0.34 2.34 6.13 0.950 3.38 795

42 1 13.41 3.84 2.12 18.8 90 2.45 2.68 0.27 1.48 4.28 0.910 3.00 1035

43 1 13.88 1.89 2.59 15.0 101 3.25 3.56 0.17 1.70 5.43 0.880 3.56 1095

44 1 13.24 3.98 2.29 17.5 103 2.64 2.63 0.32 1.66 4.36 0.820 3.00 680

45 1 13.05 1.77 2.10 17.0 107 3.00 3.00 0.28 2.03 5.04 0.880 3.35 885

46 1 14.21 4.04 2.44 18.9 111 2.85 2.65 0.30 1.25 5.24 0.870 3.33 1080

47 1 14.38 3.59 2.28 16.0 102 3.25 3.17 0.27 2.19 4.90 1.040 3.44 1065

48 1 13.90 1.68 2.12 16.0 101 3.10 3.39 0.21 2.14 6.10 0.910 3.33 985

49 1 14.10 2.02 2.40 18.8 103 2.75 2.92 0.32 2.38 6.20 1.070 2.75 1060

50 1 13.94 1.73 2.27 17.4 108 2.88 3.54 0.32 2.08 8.90 1.120 3.10 1260

51 1 13.05 1.73 2.04 12.4 92 2.72 3.27 0.17 2.91 7.20 1.120 2.91 1150

52 1 13.83 1.65 2.60 17.2 94 2.45 2.99 0.22 2.29 5.60 1.240 3.37 1265

53 1 13.82 1.75 2.42 14.0 111 3.88 3.74 0.32 1.87 7.05 1.010 3.26 1190

54 1 13.77 1.90 2.68 17.1 115 3.00 2.79 0.39 1.68 6.30 1.130 2.93 1375

55 1 13.74 1.67 2.25 16.4 118 2.60 2.90 0.21 1.62 5.85 0.920 3.20 1060

56 1 13.56 1.73 2.46 20.5 116 2.96 2.78 0.20 2.45 6.25 0.980 3.03 1120

57 1 14.22 1.70 2.30 16.3 118 3.20 3.00 0.26 2.03 6.38 0.940 3.31 970

58 1 13.29 1.97 2.68 16.8 102 3.00 3.23 0.31 1.66 6.00 1.070 2.84 1270

59 1 13.72 1.43 2.50 16.7 108 3.40 3.67 0.19 2.04 6.80 0.890 2.87 1285

60 2 12.37 0.94 1.36 10.6 88 1.98 0.57 0.28 0.42 1.95 1.050 1.82 520

61 2 12.33 1.10 2.28 16.0 101 2.05 1.09 0.63 0.41 3.27 1.250 1.67 680

62 2 12.64 1.36 2.02 16.8 100 2.02 1.41 0.53 0.62 5.75 0.980 1.59 450

63 2 13.67 1.25 1.92 18.0 94 2.10 1.79 0.32 0.73 3.80 1.230 2.46 630

64 2 12.37 1.13 2.16 19.0 87 3.50 3.10 0.19 1.87 4.45 1.220 2.87 420

65 2 12.17 1.45 2.53 19.0 104 1.89 1.75 0.45 1.03 2.95 1.450 2.23 355

66 2 12.37 1.21 2.56 18.1 98 2.42 2.65 0.37 2.08 4.60 1.190 2.30 678

67 2 13.11 1.01 1.70 15.0 78 2.98 3.18 0.26 2.28 5.30 1.120 3.18 502

68 2 12.37 1.17 1.92 19.6 78 2.11 2.00 0.27 1.04 4.68 1.120 3.48 510

69 2 13.34 0.94 2.36 17.0 110 2.53 1.30 0.55 0.42 3.17 1.020 1.93 750

70 2 12.21 1.19 1.75 16.8 151 1.85 1.28 0.14 2.50 2.85 1.280 3.07 718

71 2 12.29 1.61 2.21 20.4 103 1.10 1.02 0.37 1.46 3.05 0.906 1.82 870

[ reached 'max' / getOption("max.print") -- omitted 107 rows ]

> colnames(wine) <- c("Class", "Alcohol","Malic acid","Ash","Alcalinity of ash","Magnesium","Total phenols","Flavanoids", "Nonflavanoid phenols","Proanthocyanins","Color intensity","Hue","OD280/OD315 of diluted wines","Proline")

> # Select predictors (adjusting to specific columns if needed)

> predictors <- wine[, c(2:14)] # Modify if needed

> # Standardize data (mean = 0, variance = 1)

> predictors\_scaled <- scale(predictors)

> # Run PCA

> pca\_result <- prcomp(predictors\_scaled, center = TRUE, scale. = TRUE)

> # Summary of PCA: Importance of each principal component

> summary(pca\_result)

Importance of components:

PC1 PC2 PC3 PC4 PC5 PC6 PC7 PC8 PC9

Standard deviation 2.169 1.5802 1.2025 0.95863 0.92370 0.80103 0.74231 0.59034 0.53748

Proportion of Variance 0.362 0.1921 0.1112 0.07069 0.06563 0.04936 0.04239 0.02681 0.02222

Cumulative Proportion 0.362 0.5541 0.6653 0.73599 0.80162 0.85098 0.89337 0.92018 0.94240

PC10 PC11 PC12 PC13

Standard deviation 0.5009 0.47517 0.41082 0.32152

Proportion of Variance 0.0193 0.01737 0.01298 0.00795

Cumulative Proportion 0.9617 0.97907 0.99205 1.00000

> # Scree plot (explained variance)

> scree\_plot <- data.frame(PC = 1:length(pca\_result$sdev),

+ Variance\_Explained = (pca\_result$sdev^2 / sum(pca\_result$sdev^2)) \* 100)

> ggplot(scree\_plot, aes(x = PC, y = Variance\_Explained)) +

+ geom\_bar(stat = "identity", fill = "steelblue") +

+ geom\_line(aes(group = 1), color = "red") +

+ geom\_point(color = "red") +

+ labs(title = "Scree Plot: Variance Explained by Principal Components",

+ x = "Principal Component",

+ y = "Variance Explained (%)") +

+ theme\_minimal()

> wine$Class <- as.factor(wine$Class) # Convert to factor before plotting

> autoplot(pca\_result, data = wine, colour = 'Class', # Uses distinct colors for classes

+ loadings = TRUE,

+ loadings.colour = 'orange', # Arrows in black

+ loadings.label = TRUE,

+ loadings.label.colour = 'black', # Text in black

+ loadings.label.size = 3) +

+ scale\_color\_brewer(palette = "Set1") # Distinct colors for classes

> # Select specific predictor columns

> predictors\_reduced <- wine[, c("Alcalinity of ash", "Nonflavanoid phenols", "Flavanoids", "Proanthocyanins", "Total phenols")]

> # Run PCA

> pca\_result <- prcomp(predictors\_reduced, center = TRUE, scale. = TRUE) # Scaling applied

> # Summary of PCA: Importance of each principal component

> summary(pca\_result)

Importance of components:

PC1 PC2 PC3 PC4 PC5

Standard deviation 1.7223 0.9453 0.7688 0.64969 0.35631

Proportion of Variance 0.5933 0.1787 0.1182 0.08442 0.02539

Cumulative Proportion 0.5933 0.7720 0.8902 0.97461 1.00000

> # Scree plot (explained variance)

> scree\_plot <- data.frame(PC = 1:length(pca\_result$sdev),

+ Variance\_Explained = (pca\_result$sdev^2 / sum(pca\_result$sdev^2)) \* 100)

> ggplot(scree\_plot, aes(x = PC, y = Variance\_Explained)) +

+ geom\_bar(stat = "identity", fill = "steelblue") +

+ geom\_line(aes(group = 1), color = "red") +

+ geom\_point(color = "red") +

+ labs(title = "Scree Plot: Variance Explained by Principal Components",

+ x = "Principal Component",

+ y = "Variance Explained (%)") +

+ theme\_minimal()

> wine$Class <- as.factor(wine$Class) # Convert to factor before plotting

> autoplot(pca\_result, data = wine, colour = 'Class', # Uses distinct colors for classes

+ loadings = TRUE,

+ loadings.colour = 'orange', # Arrows in orange

+ loadings.label = TRUE,

+ loadings.label.colour = 'black', # Text in black

+ loadings.label.size = 3) +

+ scale\_color\_brewer(palette = "Set1") # Distinct colors for classes

> # Convert Class to a factor

> wine$Class <- as.factor(wine$Class)

> # Select predictors

> kNN\_predictors <- wine[, c(2:14)] # Replace with actual column names

> # Normalize predictors (scaling to [0,1] range)

> normalize <- function(x) { (x - min(x)) / (max(x) - min(x)) }

> kNN\_predictors\_norm <- as.data.frame(lapply(kNN\_predictors, normalize))

> # Add Class column back

> wine\_norm <- cbind(kNN\_predictors\_norm, Class = wine$Class)

> # Split data into training (80%) and testing (20%)

> set.seed(123) # For reproducibility

> train\_index <- createDataPartition(wine\_norm$Class, p = 0.8, list = FALSE)

> train\_data <- wine\_norm[train\_index, ]

> test\_data <- wine\_norm[-train\_index, ]

> # Define predictor and target variables

> train\_X <- train\_data[, -ncol(train\_data)] # Exclude Class

> train\_Y <- train\_data$Class # Target variable

> test\_X <- test\_data[, -ncol(test\_data)] # Exclude Class

> test\_Y <- test\_data$Class # Target variable

> # Train kNN model (choose k, e.g., k = 5)

> k\_value <- 5

> predictions <- knn(train = train\_X, test = test\_X, cl = train\_Y, k = k\_value)

> # Confusion Matrix

> confusion\_matrix <- table(Predicted = predictions, Actual = test\_Y)

> print(confusion\_matrix)

Actual

Predicted 1 2 3

1 11 1 0

2 0 13 0

3 0 0 9

> # Calculate accuracy

> accuracy <- sum(diag(confusion\_matrix)) / sum(confusion\_matrix)

> cat("Model Accuracy:", accuracy \* 100, "%\n")

Model Accuracy: 97.05882 %

> # Get performance metrics using confusionMatrix from the caret package

> conf\_matrix <- caret::confusionMatrix(predictions, test\_Y)

> # Precision, Recall, and F1 Score for each class

> cat("Precision for each class:\n")

Precision for each class:

> print(conf\_matrix$byClass[, "Precision"]) # Use matrix indexing

Class: 1 Class: 2 Class: 3

0.9166667 1.0000000 1.0000000

> cat("\nRecall for each class:\n")

Recall for each class:

> print(conf\_matrix$byClass[, "Recall"]) # Use matrix indexing

Class: 1 Class: 2 Class: 3

1.0000000 0.9285714 1.0000000

> cat("\nF1 Score for each class:\n")

F1 Score for each class:

> print(conf\_matrix$byClass[, "F1"]) # Use matrix indexing

Class: 1 Class: 2 Class: 3

0.9565217 0.9629630 1.0000000

> # Optionally, calculate the average Precision, Recall, and F1 Score

> cat("\nAverage Precision:", mean(conf\_matrix$byClass[, "Precision"]), "\n")

Average Precision: 0.9722222

> cat("Average Recall:", mean(conf\_matrix$byClass[, "Recall"]), "\n")

Average Recall: 0.9761905

> cat("Average F1 Score:", mean(conf\_matrix$byClass[, "F1"]), "\n")

Average F1 Score: 0.9731616

> # Select predictor columns

> kNN\_predictors <- wine[, c(2:14)]

> # Run PCA

> pca\_result <- prcomp(kNN\_predictors, center = TRUE, scale. = TRUE)

> # Extract the first 3 PCs

> pcs <- pca\_result$x[, 1:3] # The first 3 principal components

> # Convert pcs (PCA results) to a data frame

> pcs\_df <- as.data.frame(pcs)

> # Add the Class column for classification

> wine\_pca <- cbind(pcs\_df, Class = wine$Class)

> # Split data into training (80%) and testing (20%)

> set.seed(123) # For reproducibility

> train\_index <- createDataPartition(wine\_pca$Class, p = 0.8, list = FALSE)

> train\_data <- wine\_pca[train\_index, ]

> test\_data <- wine\_pca[-train\_index, ]

> # Define predictor and target variables

> train\_X <- train\_data[, -ncol(train\_data)] # Exclude Class

> train\_Y <- train\_data$Class # Target variable

> test\_X <- test\_data[, -ncol(test\_data)] # Exclude Class

> test\_Y <- test\_data$Class # Target variable

> # Train kNN model (choose k, e.g., k = 5)

> k\_value <- 5

> predictions <- knn(train = train\_X, test = test\_X, cl = train\_Y, k = k\_value)

> # Confusion Matrix

> confusion\_matrix <- table(Predicted = predictions, Actual = test\_Y)

> print(confusion\_matrix)

Actual

Predicted 1 2 3

1 11 2 0

2 0 12 0

3 0 0 9

> # Calculate accuracy

> accuracy <- sum(diag(confusion\_matrix)) / sum(confusion\_matrix)

> cat("Model Accuracy:", accuracy \* 100, "%\n")

Model Accuracy: 94.11765 %

> # Get performance metrics using confusionMatrix from the caret package

> conf\_matrix <- caret::confusionMatrix(predictions, test\_Y)

> # Precision, Recall, and F1 Score for each class

> cat("Precision for each class:\n")

Precision for each class:

> print(conf\_matrix$byClass[, "Precision"]) # Use matrix indexing

Class: 1 Class: 2 Class: 3

0.8461538 1.0000000 1.0000000

> cat("\nRecall for each class:\n")

Recall for each class:

> print(conf\_matrix$byClass[, "Recall"]) # Use matrix indexing

Class: 1 Class: 2 Class: 3

1.0000000 0.8571429 1.0000000

> cat("\nF1 Score for each class:\n")

F1 Score for each class:

> print(conf\_matrix$byClass[, "F1"]) # Use matrix indexing

Class: 1 Class: 2 Class: 3

0.9166667 0.9230769 1.0000000

> # Optionally, calculate the average Precision, Recall, and F1 Score

> cat("\nAverage Precision:", mean(conf\_matrix$byClass[, "Precision"]), "\n")

Average Precision: 0.9487179

> cat("Average Recall:", mean(conf\_matrix$byClass[, "Recall"]), "\n")

Average Recall: 0.952381

> cat("Average F1 Score:", mean(conf\_matrix$byClass[, "F1"]), "\n")

Average F1 Score: 0.9465812