|  |
| --- |
| > ##########################################  > ### Principal Component Analysis (PCA) ###  > ##########################################  > library(ggplot2)  > library(readr)  > library(ggfortify)  > library(e1071)  > library(class)  > library(psych)  > # PCA with iris dataset  > wine <- read\_csv(unz("C:/Users/chaos/Downloads/wine.zip", "wine.data"), col\_names = FALSE)  **Rows:** 178 **Columns:** 14  ── **Column specification** ────────────────────────────────────────────────────────────────────────────────  **Delimiter:** ","  dbl (14): X1, X2, X3, X4, X5, X6, X7, X8, X9, X10, X11, X12, X13, X14  ℹ Use `spec()` to retrieve the full column specification for this data.  ℹ Specify the column types or set `show\_col\_types = FALSE` to quiet this message.  > names(wine) <- c("Type","Alcohol","Malic acid","Ash","Alcalinity of ash","Magnesium","Total phenols","Flavanoids","Nonflavanoid Phenols","Proanthocyanins","Color Intensity","Hue","Od280/od315 of diluted wines","Proline")  > head(wine)  # A tibble: 6 × 14  Type Alcohol `Malic acid` Ash `Alcalinity of ash` Magnesium `Total phenols` Flavanoids  *<dbl>* *<dbl>* *<dbl>* *<dbl>* *<dbl>* *<dbl>* *<dbl>* *<dbl>*  1 1 14.2 1.71 2.43 15.6 127 2.8 3.06  2 1 13.2 1.78 2.14 11.2 100 2.65 2.76  3 1 13.2 2.36 2.67 18.6 101 2.8 3.24  4 1 14.4 1.95 2.5 16.8 113 3.85 3.49  5 1 13.2 2.59 2.87 21 118 2.8 2.69  6 1 14.2 1.76 2.45 15.2 112 3.27 3.39  # ℹ 6 more variables: `Nonflavanoid Phenols` <dbl>, Proanthocyanins <dbl>, `Color Intensity` <dbl>,  # Hue <dbl>, `Od280/od315 of diluted wines` <dbl>, Proline <dbl>  > wine$Type <- as.factor(wine$Type)  > wine <- wine[,-c(4,5,10)]  > pairs.panels(wine[,-1],gap = 0,bg = c("red", "yellow", "blue")[wine$Type],pch=21)  > # Performing PCA with scaling variables  > wine.pca <- prcomp(wine[, -1], center = TRUE, scale. = TRUE)  > # Plotting the first two principal components  > autoplot(wine.pca, data = wine, colour = 'Type', x = 1, y = 2,  + loadings = TRUE, loadings.colour = "blue",  + loadings.label = TRUE, loadings.label.size = 3) +  + ggtitle("Wine PCA: First and Second Principal Components") +  + theme\_minimal()  > # Identifying the variables that contribute the most to the 1st PC in rstudio  > # Loadings for the first principal component (PC1)  > loading\_scores <- wine.pca$rotation[, 1]  > # Calculating absolute values of the loading scores  > abs\_loading\_scores <- abs(loading\_scores)  > ranked\_variables <- sort(abs\_loading\_scores, decreasing = TRUE)  > print(ranked\_variables)  Flavanoids Total phenols Od280/od315 of diluted wines  0.4555529 0.4234687 0.4166142  Hue Nonflavanoid Phenols Proline  0.3392301 0.3215373 0.2968863  Malic acid Magnesium Alcohol  0.2774631 0.1482252 0.1387659  Color Intensity  0.1214953  > # Train a classifier model to predict wine type using the 11 attributes.  > set.seed(42)  > # Sample ~70% of the dataset  > sample\_index <- sample(nrow(wine), 0.7 \* nrow(wine))  > wine.train <- wine[sample\_index, ]  > wine.test <- wine[-sample\_index, ]  > # Extract predictor variables and target variable for training and testing  > train\_x <- wine.train[, -1]  > train\_y <- wine.train$Type  > test\_x <- wine.test[, -1]  > test\_y <- wine.test$Type  > # Calculate the approximate square root  > sqrt\_train\_size <- sqrt(nrow(wine.train))  > k <- round(sqrt\_train\_size)  > # Choosing the number of neighbors (k) for the model  > accuracy <- c()  > ks <- c(3, 6, 9, 12, 15, 18, 21, 24, 27, 30, 33, 36, 39, 42, 45, 48, 51, 54, 57, 60, 63, 66, 69, 72, 75, 78, 81, 84, 87, 90, 93, 96, 99)  > # Looping through different k values to evaluate accuracy  > for (k in ks) {  + # Train and predict using k-NN  + KNNpred <- knn(train = train\_x, test = test\_x, cl = train\_y, k = k)  +  + # Create confusion matrix  + contingency.table <- table(Predicted = KNNpred, Actual = test\_y)  + contingency.matrix <- as.matrix(contingency.table)  +  + # Calculate accuracy for each k  + accuracy <- c(accuracy, sum(diag(contingency.matrix)) / length(test\_y))  + }  > # Plot accuracy against k values  > plot(ks, accuracy, type = "b", xlab = "Number of Neighbors (k)", ylab = "Accuracy", main = "k-NN Accuracy for Different k Values", ylim = c(min(accuracy), max(accuracy)))  > # Selecting the best k (based on highest accuracy) and print metrics  > best\_k <- ks[which.max(accuracy)]  > cat("Best k:", best\_k, "\n")  Best k: 27  > # Train and predict using the best k  > KNNpred <- knn(train = train\_x, test = test\_x, cl = train\_y, k = best\_k)  > # Printing  > knn\_confusion\_matrix <- table(Predicted = KNNpred, Actual = test\_y)  > print("Confusion Matrix for best k:")  [1] "Confusion Matrix for best k:"  > print(knn\_confusion\_matrix)  Actual  Predicted 1 2 3  1 19 1 2  2 0 15 3  3 2 3 9  > # Computing precision, recall, and F1 score  > compute\_metrics <- function(conf\_matrix) {  + diag <- diag(conf\_matrix)  + rowsums <- apply(conf\_matrix, 1, sum)  + colsums <- apply(conf\_matrix, 2, sum)  + precision <- diag / colsums  + recall <- diag / rowsums  + f1 <- 2 \* (precision \* recall) / (precision + recall)  + data.frame(Precision = precision, Recall = recall, F1 = f1)  + }  > # Print  > print("k-NN Metrics for best k:")  [1] "k-NN Metrics for best k:"  > knn\_metrics <- compute\_metrics(knn\_confusion\_matrix)  > print(knn\_metrics)  Precision Recall F1  1 0.9047619 0.8636364 0.8837209  2 0.7894737 0.8333333 0.8108108  3 0.6428571 0.6428571 0.6428571  > #Train a classifier model to predict wine type using the data projected into the first 3 PCs.  > set.seed(36)  > wine\_pca\_data <- data.frame(Type = wine$Type, wine.pca$x[, 1:3])  > # Sample ~70% of the PCA-transformed data  > sample\_index <- sample(nrow(wine\_pca\_data), 0.7 \* nrow(wine\_pca\_data))  > train\_pca\_data <- wine\_pca\_data[sample\_index, ]  > test\_pca\_data <- wine\_pca\_data[-sample\_index, ]  > # Extract predictor variables and target variable for training and testing  > train\_pca\_x <- train\_pca\_data[, -1]  > train\_pca\_y <- train\_pca\_data$Type  > test\_pca\_x <- test\_pca\_data[, -1]  > test\_pca\_y <- test\_pca\_data$Type  > # k tests  > accuracy <- c()  > ks <- c(3, 6, 9, 12, 15, 18, 21, 24, 27, 30, 33, 36, 39, 42, 45, 48, 51, 54, 57, 60, 63, 66, 69, 72, 75, 78, 81, 84, 87, 90, 93, 96, 99)  > # Loop through different k values to evaluate accuracy on PCA-reduced data  > for (k in ks) {  + # Train and predict using k-NN  + knn\_pca\_pred <- knn(train = train\_pca\_x, test = test\_pca\_x, cl = train\_pca\_y, k = k)  +  + # Create confusion matrix  + pca\_contingency\_table <- table(Predicted = knn\_pca\_pred, Actual = test\_pca\_y)  + pca\_contingency\_matrix <- as.matrix(pca\_contingency\_table)  +  + # Calculate accuracy for each k  + accuracy <- c(accuracy, sum(diag(pca\_contingency\_matrix)) / length(test\_pca\_y))  + }  > # Plot accuracy against k values  > plot(ks, accuracy, type = "b", xlab = "Number of Neighbors (k)", ylab = "Accuracy", main = "k-NN Accuracy on PCA Data for Different k Values", ylim = c(min(accuracy), max(accuracy)))  > # Select the best k (based on highest accuracy) and display metrics  > best\_k <- ks[which.max(accuracy)]  > cat("Best k:", best\_k, "\n")  Best k: 3  > # Train and predict using the best k on PCA-reduced data  > knn\_pca\_pred <- knn(train = train\_pca\_x, test = test\_pca\_x, cl = train\_pca\_y, k = best\_k)  > # Display the final confusion matrix for the best k on PCA data  > knn\_pca\_confusion\_matrix <- table(Predicted = knn\_pca\_pred, Actual = test\_pca\_y)  > print("Confusion Matrix for best k (PCA data):")  [1] "Confusion Matrix for best k (PCA data):"  > print(knn\_pca\_confusion\_matrix)  Actual  Predicted 1 2 3  1 18 0 0  2 0 19 0  3 0 0 17  > # Computing precision, recall, and F1 score  > compute\_metrics <- function(conf\_matrix) {  + diag <- diag(conf\_matrix)  + rowsums <- apply(conf\_matrix, 1, sum)  + colsums <- apply(conf\_matrix, 2, sum)  + precision <- diag / colsums  + recall <- diag / rowsums  + f1 <- 2 \* (precision \* recall) / (precision + recall)  + data.frame(Precision = precision, Recall = recall, F1 = f1)  + }  > # Print  > print("k-NN Metrics for best k (PCA-transformed data):")  [1] "k-NN Metrics for best k (PCA-transformed data):"  > knn\_pca\_metrics <- compute\_metrics(knn\_pca\_confusion\_matrix)  > print(knn\_pca\_metrics)  Precision Recall F1  1 1 1 1  2 1 1 1  3 1 1 1  > # Drop the least contributing variables (Color Intensity, Alcohol, Magnesium)  > wine\_reduced <- wine[, !names(wine) %in% c("Color Intensity", "Alcohol", "Magnesium")]  > # Perform PCA again on the reduced dataset  > wine\_reduced.pca <- prcomp(wine\_reduced[, -1], center = TRUE, scale. = TRUE)  > # Plotting the first two principal components  > autoplot(wine\_reduced.pca, data = wine, colour = 'Type', x = 1, y = 2,  + loadings = TRUE, loadings.colour = "blue",  + loadings.label = TRUE, loadings.label.size = 3) +  + ggtitle("Reduced Wine Dataset PCA: First and Second Principal Components") +  + theme\_minimal()  > # Rerunning PCA: Reduced dataset onto the first three principal components  > set.seed(60)  > wine\_reduced\_pca\_data <- data.frame(Type = wine$Type, wine\_reduced.pca$x[, 1:3])  > # Sample ~70% of the reduced PCA-transformed data  > sample\_index <- sample(nrow(wine\_reduced\_pca\_data), 0.7 \* nrow(wine\_reduced\_pca\_data))  > train\_reduced\_pca\_data <- wine\_reduced\_pca\_data[sample\_index, ]  > test\_reduced\_pca\_data <- wine\_reduced\_pca\_data[-sample\_index, ]  > # Extracting predictor variables and target variable for training and testing  > train\_reduced\_pca\_x <- train\_reduced\_pca\_data[, -1]  > train\_reduced\_pca\_y <- train\_reduced\_pca\_data$Type  > test\_reduced\_pca\_x <- test\_reduced\_pca\_data[, -1]  > test\_reduced\_pca\_y <- test\_reduced\_pca\_data$Type  > # k test  > accuracy <- c()  > ks <- c(3, 6, 9, 12, 15, 18, 21, 24, 27, 30, 33, 36, 39, 42, 45, 48, 51, 54, 57, 60, 63, 66, 69, 72, 75, 78, 81, 84, 87, 90, 93, 96, 99)  > # Loop through different k values on reduced PCA data  > for (k in ks) {  + # Train and predict using k-NN on the reduced PCA-transformed data  + knn\_reduced\_pca\_pred <- knn(train = train\_reduced\_pca\_x, test = test\_reduced\_pca\_x, cl = train\_reduced\_pca\_y, k = k)  +  + # Create confusion matrix  + reduced\_pca\_contingency\_table <- table(Predicted = knn\_reduced\_pca\_pred, Actual = test\_reduced\_pca\_y)  + reduced\_pca\_contingency\_matrix <- as.matrix(reduced\_pca\_contingency\_table)  +  + # Calculate accuracy for each k  + accuracy <- c(accuracy, sum(diag(reduced\_pca\_contingency\_matrix)) / length(test\_reduced\_pca\_y))  + }  > # Plot accuracy against k values  > plot(ks, accuracy, type = "b", xlab = "Number of Neighbors (k)", ylab = "Accuracy", main = "k-NN Accuracy on Reduced PCA Data for Different k Values", ylim = c(min(accuracy), max(accuracy)))  > # Select the best k (based on highest accuracy) and display metrics  > best\_k <- ks[which.max(accuracy)]  > cat("Best k:", best\_k, "\n")  Best k: 18  > # Train and predict using the best k on reduced PCA data  > knn\_reduced\_pca\_pred <- knn(train = train\_reduced\_pca\_x, test = test\_reduced\_pca\_x, cl = train\_reduced\_pca\_y, k = best\_k)  > # Print  > knn\_reduced\_pca\_confusion\_matrix <- table(Predicted = knn\_reduced\_pca\_pred, Actual = test\_reduced\_pca\_y)  > print("Confusion Matrix for best k (Reduced PCA data):")  [1] "Confusion Matrix for best k (Reduced PCA data):"  > print(knn\_reduced\_pca\_confusion\_matrix)  Actual  Predicted 1 2 3  1 22 2 0  2 2 11 0  3 0 2 15  > # Compute precision, recall, and F1 score  > compute\_metrics <- function(conf\_matrix) {  + diag <- diag(conf\_matrix)  + rowsums <- apply(conf\_matrix, 1, sum)  + colsums <- apply(conf\_matrix, 2, sum)  + precision <- diag / colsums  + recall <- diag / rowsums  + f1 <- 2 \* (precision \* recall) / (precision + recall)  + data.frame(Precision = precision, Recall = recall, F1 = f1)  + }  > # Print  > print("k-NN Metrics for best k (Reduced PCA-transformed data):")  [1] "k-NN Metrics for best k (Reduced PCA-transformed data):"  > knn\_reduced\_pca\_metrics <- compute\_metrics(knn\_reduced\_pca\_confusion\_matrix)  > print(knn\_reduced\_pca\_metrics)  Precision Recall F1  1 0.9166667 0.9166667 0.9166667  2 0.7333333 0.8461538 0.7857143  3 1.0000000 0.8823529 0.9375000  >  > ##Comparing the 3 classification models-- see contingency tables/recall/precision/f1 metrics. |
|  |
| |  | | --- | | > | |