install.packages(“ggplot2”)— title: “CARET\_Lab” output: html\_document date: “2023-05-24” —

Package loading

library(caret)

## Warning: package 'caret' was built under R version 4.2.3

## Loading required package: ggplot2

## Warning: package 'ggplot2' was built under R version 4.2.3

## Loading required package: lattice

library(lattice)

Load Data

# attach the iris dataset to the environment  
data(iris)  
# rename the dataset  
dataset <- iris

Task1: Create a Validation/Training Dataset You need to split the loaded dataset into two, 80% of which we will use to train our models and 20% that we will hold back as a validation dataset. Hint: use createDataPartition function

# Create the training and test datasets  
set.seed(100)  
# Step 1: Get row numbers for the training data  
trainRowNumbers <- createDataPartition(dataset$Species, p=0.8, list=FALSE)  
# Step 2: Create the training dataset  
trainData <- dataset[trainRowNumbers,]  
# Step 3: Create the test dataset  
testData <- dataset[-trainRowNumbers,]

Task2: Summarize Dataset Use skimr library to summarize the dataset

library(skimr)

## Warning: package 'skimr' was built under R version 4.2.3

skimmed<-skim\_to\_wide(trainData)

## Warning: 'skim\_to\_wide' is deprecated.  
## Use 'skim()' instead.  
## See help("Deprecated")

## Warning: 'skim\_to\_wide' is deprecated.  
## Use 'skim()' instead.  
## See help("Deprecated")  
skimmed

Data summary

|  |  |
| --- | --- |
| Name | Piped data |
| Number of rows | 120 |
| Number of columns | 5 |
| \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ |  |
| Column type frequency: |  |
| factor | 1 |
| numeric | 4 |
| \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ |  |
| Group variables | None |

**Variable type: factor**

| skim\_variable | n\_missing | complete\_rate | ordered | n\_unique | top\_counts |
| --- | --- | --- | --- | --- | --- |
| Species | 0 | 1 | FALSE | 3 | set: 40, ver: 40, vir: 40 |

**Variable type: numeric**

| skim\_variable | n\_missing | complete\_rate | mean | sd | p0 | p25 | p50 | p75 | p100 | hist |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| Sepal.Length | 0 | 1 | 5.86 | 0.82 | 4.3 | 5.1 | 5.8 | 6.4 | 7.7 | ▃▇▇▅▂ |
| Sepal.Width | 0 | 1 | 3.07 | 0.43 | 2.2 | 2.8 | 3.0 | 3.4 | 4.4 | ▃▇▇▂▁ |
| Petal.Length | 0 | 1 | 3.77 | 1.78 | 1.0 | 1.6 | 4.4 | 5.1 | 6.9 | ▇▁▅▇▂ |
| Petal.Width | 0 | 1 | 1.20 | 0.77 | 0.1 | 0.3 | 1.3 | 1.8 | 2.5 | ▇▁▇▅▃ |

Task3: split input and output It is the time to seperate the input attributes and the output attributes. call the inputs attributes x and the output attribute (or class) y.

x = trainData[,1:4]  
y = trainData[,5]

Task4: Train Control for Validation Test

We will use 10-fold crossvalidation to estimate accuracy.

# Run algorithms using 10-fold cross validation  
control <- trainControl(method="cv", number=10)  
metric <- "Accuracy"

Task5: Model Training Train 5 different algorithms using ‘train’ function:

* Linear Discriminant Analysis (LDA)
* Classification and Regression Trees (CART).
* k-Nearest Neighbors (kNN).
* Support Vector Machines (SVM) with a linear kernel.
* Random Forest (RF)

# Set the seed for reproducibility  
set.seed(100)  
# Train the model using LDA on the training data itself.  
lda\_model = train(Species ~ ., data=trainData, method='lda',   
trControl=control, metric=metric)  
#---------------------------------------------------------------------------  
# Set the seed for reproducibility  
set.seed(100)  
# Train the model using rpart on the training data itself.  
rpart\_model = train(Species ~ ., data=trainData, method='rpart',   
trControl=control, metric=metric)  
#---------------------------------------------------------------------------  
# Set the seed for reproducibility  
set.seed(100)  
# Train the model using kNN on the training data itself.  
knn\_model = train(Species ~ ., data=trainData, method='knn',   
trControl=control, metric=metric)  
#---------------------------------------------------------------------------  
# Set the seed for reproducibility  
set.seed(100)  
# Train the model using svmRadial on the training data itself.  
svm\_model = train(Species ~ ., data=trainData, method='svmRadial',   
trControl=control, metric=metric)  
#---------------------------------------------------------------------------  
# Set the seed for reproducibility  
set.seed(100)  
# Train the model using rf on the training data itself.  
rf\_model = train(Species ~ ., data=trainData, method='rf', trControl=control,   
metric=metric)

Task6: Select the Best Model We now have 5 models and accuracy estimations for each. We need to compare the models to each other and select the most accurate. Use resamples function to complete this task

# Compare model performances using resample()  
models\_compare<-resamples(list(lda=lda\_model, rpart=rpart\_model,   
knn=knn\_model, svm=svm\_model, rf=rf\_model))  
# Summary of the models performances  
summary(models\_compare)

##   
## Call:  
## summary.resamples(object = models\_compare)  
##   
## Models: lda, rpart, knn, svm, rf   
## Number of resamples: 10   
##   
## Accuracy   
## Min. 1st Qu. Median Mean 3rd Qu. Max. NA's  
## lda 0.9166667 0.9375000 1.0000000 0.9750000 1 1 0  
## rpart 0.8333333 0.8541667 0.9166667 0.9250000 1 1 0  
## knn 0.9166667 1.0000000 1.0000000 0.9833333 1 1 0  
## svm 0.8333333 0.9166667 0.9583333 0.9416667 1 1 0  
## rf 0.9166667 0.9166667 1.0000000 0.9666667 1 1 0  
##   
## Kappa   
## Min. 1st Qu. Median Mean 3rd Qu. Max. NA's  
## lda 0.875 0.90625 1.0000 0.9625 1 1 0  
## rpart 0.750 0.78125 0.8750 0.8875 1 1 0  
## knn 0.875 1.00000 1.0000 0.9750 1 1 0  
## svm 0.750 0.87500 0.9375 0.9125 1 1 0  
## rf 0.875 0.87500 1.0000 0.9500 1 1 0

What was the most accurate model?

Task7: Make Prediction (Confusion Matrix) Now we want to get an idea of the accuracy of the best model on our validation set. Use ‘predict’ and confusionMatrix functions to complete this task.

# Predict on testData using lda\_model  
predicted <- predict(lda\_model, testData)  
confusionMatrix(reference= testData$Species,data = predicted)

## Confusion Matrix and Statistics  
##   
## Reference  
## Prediction setosa versicolor virginica  
## setosa 10 0 0  
## versicolor 0 9 0  
## virginica 0 1 10  
##   
## Overall Statistics  
##   
## Accuracy : 0.9667   
## 95% CI : (0.8278, 0.9992)  
## No Information Rate : 0.3333   
## P-Value [Acc > NIR] : 2.963e-13   
##   
## Kappa : 0.95   
##   
## Mcnemar's Test P-Value : NA   
##   
## Statistics by Class:  
##   
## Class: setosa Class: versicolor Class: virginica  
## Sensitivity 1.0000 0.9000 1.0000  
## Specificity 1.0000 1.0000 0.9500  
## Pos Pred Value 1.0000 1.0000 0.9091  
## Neg Pred Value 1.0000 0.9524 1.0000  
## Prevalence 0.3333 0.3333 0.3333  
## Detection Rate 0.3333 0.3000 0.3333  
## Detection Prevalence 0.3333 0.3000 0.3667  
## Balanced Accuracy 1.0000 0.9500 0.9750