Accuracy = $\frac{\text{correct predictions}}{\text{total predictions}}$

SVM

↪ supervised learning
↪ machine learns to predict target variable

↪ to install packages
install.packages(" ")

```
library (e1071)

data (iris)

set.seed (123)
indices ← sample (1:nrow (iris), 0.8 * nrow(iris))
train_data ← iris [indices, -5]
```

- no. of rows (nrow(iris))
- random sampling 80% data (0.8 * nrow(iris))
- iris [indices, -5] → includes sampling data & excludes row 5 which is target class

```
train_labels ← iris $ species [indices]
test_data ← iris [-indices, -5]
test_labels ← iris $ species [-indices]
```

- train_labels ↪ sample labels
- test_data ↪ excludes training data and target variable
- test_labels ↪ excluded sample

```
svm_model ← svm (train_data, train_labels,
                 kernel = "radial", cost = 1)

svm_predictions ← predict (svm_model, test_data)

confusion_matrix ← table (Actual = test_labels, predictions =
                   svm_predictions)

accuracy ← sum (diag (confusion_matrix))/sum (confusion_
           matrix)

confusion_matrix
accuracy
```

unsupervised learning
no target variable
data is predicted based on majority class

KNN

```
library(class)
data(iris)

set.seed(123)
train_data indices <- sample(1:nrow(iris), size = 75 OR 0.8 * nrow(iris))
train_data <- iris[indices, ]   } -> no target variable
test_data <- iris[-indices, ]

knn_predictions <- knn(train_data[, -5], test_data[, -5],
                       train_data$species, k = 1)

confusion_matrix <- table(predicted knn_predictions, test_data$species)

accuracy <- sum(diag(confusion_matrix))/ nrow(test_data)

confusion_matrix
accuracy
```

randomly samples 75 indices

exclude target variables

no. of neighbours

confusion_matrix ← table(predicted, test_labels)

Bootstrap aggregating

Bagging

install.packages("ipred")

library(ipred)

data(iris)

set.seed(123)

multiple models are trained on different samples

ensemble model created using bagging

n.o. of bootstrap samples

bagged_model ← bagging(species ~ ., data = iris, nbagg = 50)

species predicted based on all other variables

Predictions ← predict(bagged_model, newdata = iris)

table(predictions, iris$species) → confusion_matrix

ensemble technique for weak models like decision trees

Boosting

```
library(gbm)          # gradient boosting model
data(iris)
iris$species <- as.numeric(factor(iris$species))   # gbm expects numeric values

set.seed(123)

boosted_model <- gbm(species ~ ., data = iris,
                     distribution = "multinomial", n.trees = 100,   # no. of decision trees
                     interaction.depth = 3, shrinkage = 0.1)        # max depth of each tree; learning rate

predictions <- predict(boosted_model, newdata = iris,
                       n.trees = 100, type = "response")   # predicted value type: response = probability

predicted_labels <- apply(predictions, 1, which.max)   # converts predicted probabilities into class labels by choosing the class with highest probability

table(predicted_labels, iris$species)
```

Random Forest

```
library (mlbench)
library (caret)
library (random forest)
dataset ← read.csv (" —— \\ soybean. csv)
```

#preprocessing

```
sum (is.na (dataset))        } #1
soybean ← na.omit (dataset)
```

#2 preproc ← preprocess (soybean [, -1], method = c ("center", "scale"))

↳ caret function. selects everything except column 1 and centers (subtract mean) & scales (divide by std) the data

soybean [, -1] ← predict (preproc, soybean [, -1])

↳ applies scaled values to columns (transformation)

set.seed (123)

(indices) split index ← createDataPartition (soybean $ class, p = 0.8, list = FALSE)   ← caret package; 80% training data

```
training_data ← soybean [split Index, ]
test_data ← soybean [-splitindex, ]

model ← train (class ~ ., data = training_data, method = "rf")

predictions ← predict (Model, newdata = testing _data)
prediction ← as.factor (predictions)     ← categorical data
testing_data $ class ← as.factor (test_dat $ class)
```

typeof (test_data $ class) → prints datatype of class column values

typeof (Prediction) → prints datatype of predictions variable

confusionMatrix (predictions, test_data $ class)

Decision trees

```
library (rpart)
titanic ← read.csv ("          \titanic.csv")

                c  selecting columns
titanic ← titanic[, c("   ",   "   ",   "   ")]

#preprocessing
#1     titanic ← na.omit (titanic)
#2     titanic$sex ← as.factor (titanic$sex)
#3

tree ← rpart (survived ~ , data = titanic, method = "class")

tree      #printing

library(rpart.plot)
prp(tree)
  ↳ plots decision tree

predictions ← predict (tree, newdata = titanic, type = "class")

confusion_matrix ← table ( predictions, titanic$survived)
```

Naive Bayes

```
library (MLbench)
library ( caret)
library (e1071)
dataset ← read.csv

#preprocessing

set.seed (123)
indices ← createDataPartition (soybean$ class, p=0.8, list=FALSE)
train_data ← soybean [indices, ]
test_data ← soybean [-indices, ]

set.seed (120)
classifier_soybean ← naiveBayes (class ~., data = train_data)
classifier_soybean

matrix ← confusion table(testing_data$class, y_pred)

y_pred ← predict (classifier_soybean, newdata= test_data)

cm ← table ( test_data$ class, y_pred)
cm

confusionMatrix (cm)
```

↳ evaluation