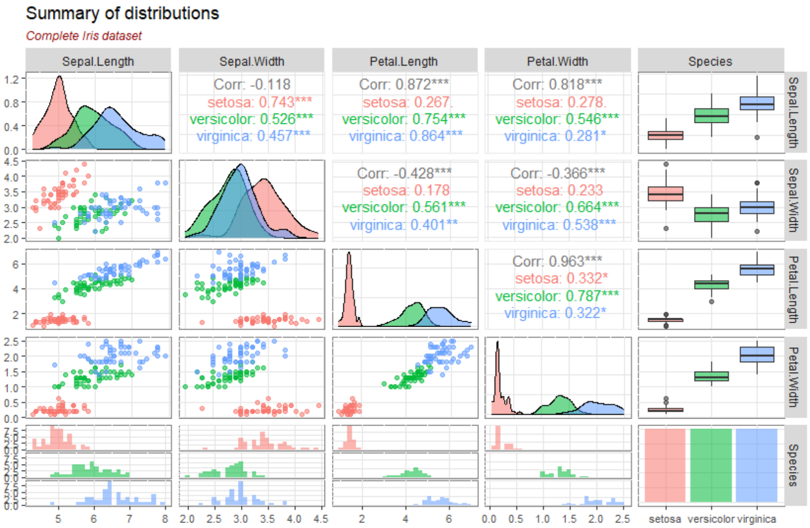
Visualization of Iris dataset



Some metrics

TP = true positive, TN = true negative, FP = false positive, FN = false negative

Accuracy. The number of samples correctly classified out of all the samples present in the (test) set.

$$Accuracy = \frac{TP + TN}{(TP + TN + FP + FN)}$$

Precision (for the positive class). The number of samples actually belonging to the positive class out of all the samples that were predicted to be of the positive class by the model.

$$Precision = \frac{TP}{(TP + FP)}$$

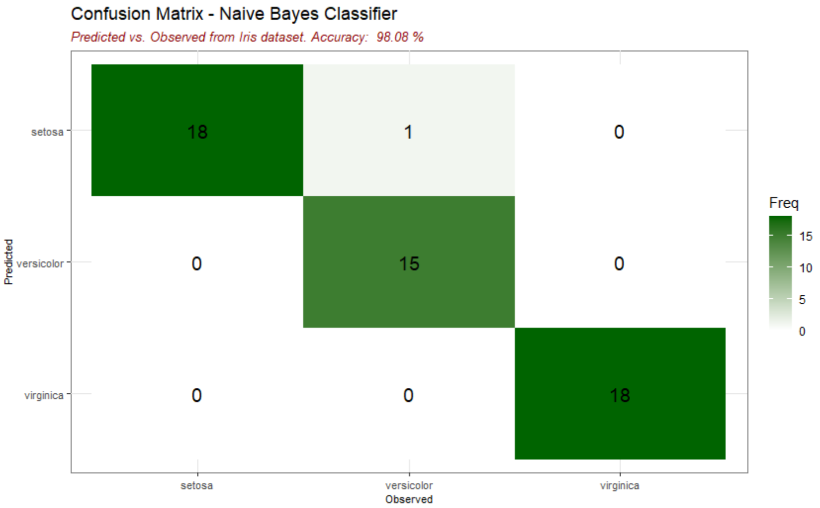
Recall (for the positive class). The number of samples predicted correctly to be belonging to the positive class out of all the samples that actually belong to the positive class.

$$Recall = \frac{TP}{(TP + FP)}$$

F1-Score (for the positive class). The harmonic mean of the precision and recall scores obtained for the positive class.

$$F1 - score = \frac{2 * Precision * Recall}{(Precision + Recall)}$$

Naïve Bayes: confusion matrix

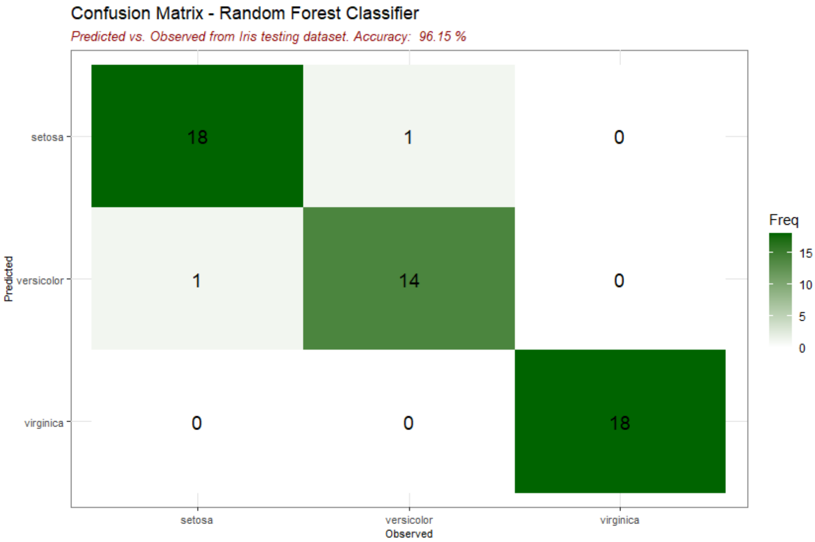


Naïve Bayes: code chunk and metrics

```r
library(klaR)

# 1. Build a Naive Bayes Classifier
set.seed(2023)
nb_model <- NaiveBayes(Species ~ ., data=training) # train Na ve Bayes model
pred_nb <- predict(nb_model, testing) # apply Na ve Bayes model on test set
pred_nb_training <- predict(nb_model, training) # apply Na ve Bayes model on train set
```

	Training	Testing
accuracy	94.90	98.08
precision	94.91	97.92
recall	94.99	98.25
f1-score	94.95	98.08

Random Forest: confusion matrix

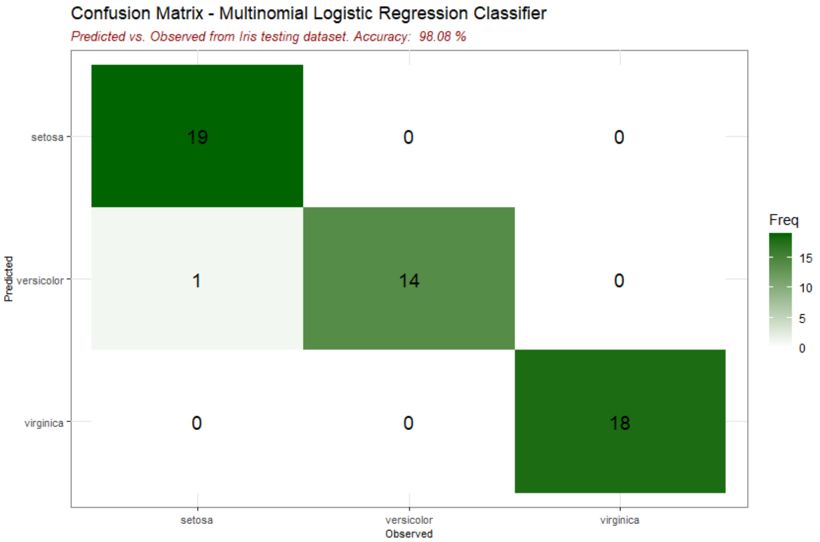


Random Forest: code chunk and metrics

```r
library(randomForest)

# 1. Build a Random Forest learning tree Classifier
set.seed(2023)
rf_model <- randomForest(Species~., data=training, ntree=100,proximity=TRUE) # train RF model
pred_rf <- predict(rf_model, testing) # apply RF model on test set
pred_rf_training <- predict(rf_model, training) # apply RF model on train set
```

	Training	Testing
accuracy	100.00	96.15
precision	100.00	96.02
recall	100.00	96.02
f1-score	100.00	96.02

Logistic Regression: confusion matrix

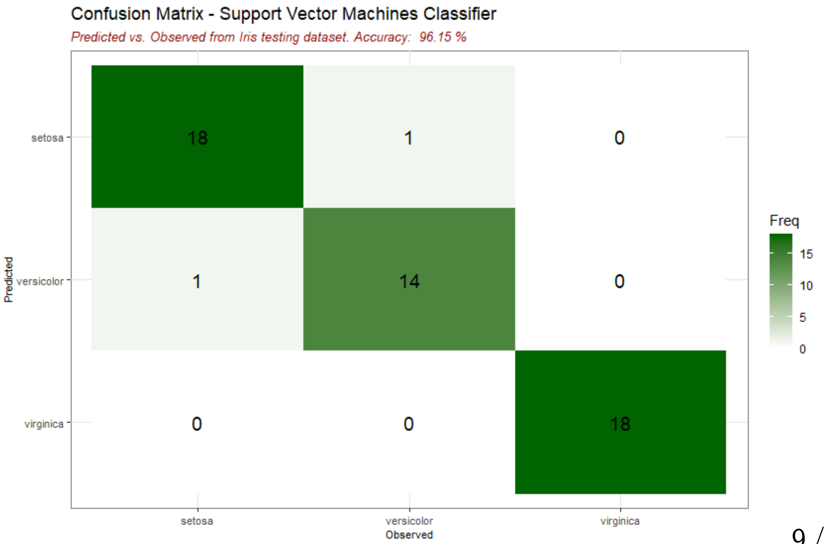


Logistic Regression: code chunk and metrics

```
library(stats4)
library(splines)
library(VGAM)

# 1. Build a Multinomial Logistic Regression Classifier
set.seed(2023)
mlr_model <- vglm(Species ~ ., family=multinomial, training)
pred_mlr_training<- predict(mlr_model, training, type = "response")
pred_mlr_testing<- predict(mlr_model, testing, type = "response")
predictions <- apply(pred_mlr_testing, 1, which.max)
predictions_training <- apply(pred_mlr_training, 1, which.max)
```

	Training	Testing
accuracy	100.00	98.08
precision	100.00	98.33
recall	100.00	97.78
f1-score	100.00	98.05

Support Vector Machines: confusion matrix

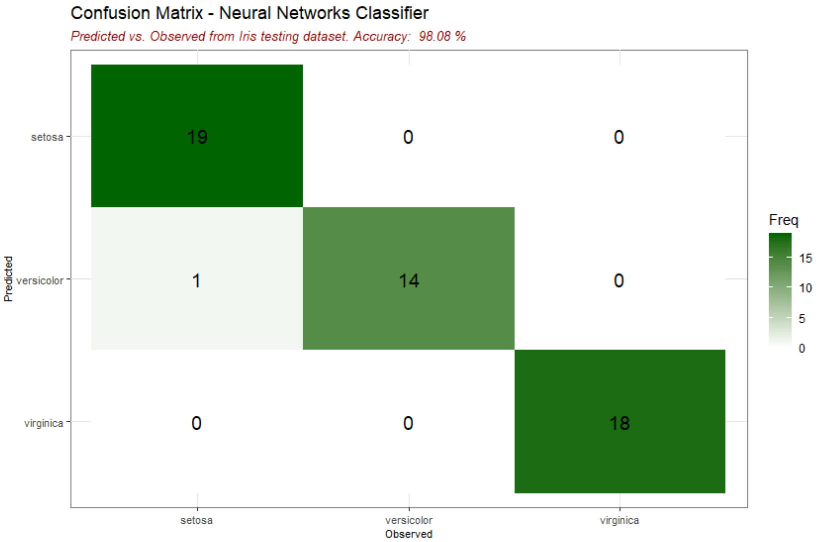


Support Vector Machines: code chunk and metrics

```r
library(e1071)

# 1. Build a Support Vector Machines Classifier
set.seed(2023)
svm_model <- svm(Species ~ ., data=training,
                 kernel="radial") #linear/polynomial/sigmoid
pred_svm <- predict(svm_model, testing)
pred_svm_training <- predict(svm_model, training) # apply svm model on train set
```

	Training	Testing
accuracy	96.94	96.15
precision	97.04	96.02
recall	96.90	96.02
f1-score	96.97	96.02

Neural Network: confusion matrix

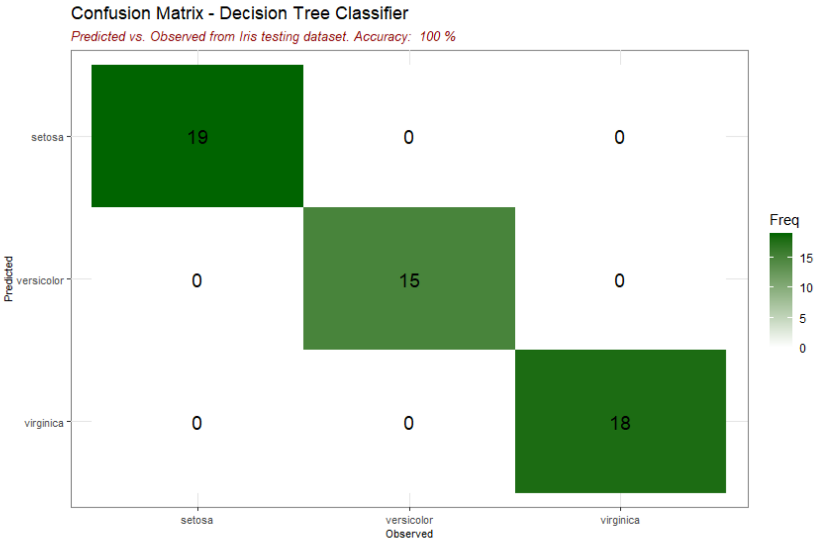


Neural Networks: code chunk and metrics

```r
library(neuralnet)

# 1. Build a Neural Network Classifier
set.seed(2023)
iris$setosa <- iris$Species=="setosa"
iris$virginica <- iris$Species == "virginica"
iris$versicolor <- iris$Species == "versicolor"

# spliting into test and training again because we added variables
ind <- sample(2, nrow(iris),replace=TRUE,prob=c(0.7,0.3))
training <- iris[ind==1,] ; testing <- iris[ind==2,]

nn_model <- neuralnet(setosa+versicolor+virginica ~
                        Sepal.Length + Sepal.Width + Petal.Length + Petal.Width,
                      data=training, hidden=c(10,10), rep = 5, err.fct = "ce",
                      linear.output = F, lifesign = "minimal",
                      stepmax = 1000000,  threshold = 0.001)
```

	Training	Testing
accuracy	100.00	98.08
precision	100.00	98.33
recall	100.00	97.78
f1-score	100.00	98.05

Decision Tree: confusion matrix

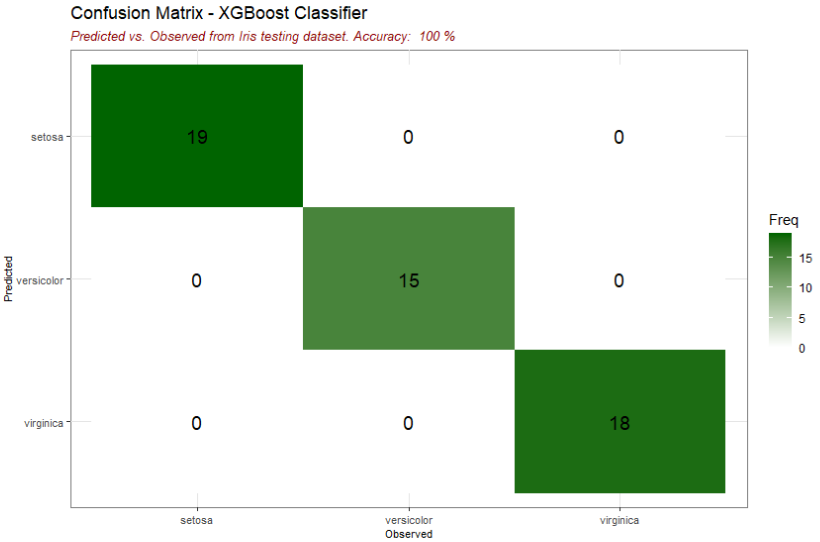


Decision Tree: code chunk and metrics

```r
library(rpart)

# 1. Build a Decision Tree Classifier
set.seed(2023)
dt_model <- rpart(Species ~.,
                  data = training,
                  method = "class",
                  control = rpart.control(cp = 0),
                  parms = list(split = "information"))

pred_dt_training<- predict(dt_model, training, type = "class")
pred_dt_testing<- predict(dt_model, testing, type = "class")
```

	Training	Testing
accuracy	100.00	100.00
precision	100.00	100.00
recall	100.00	100.00
f1-score	100.00	100.00

XGBoost: confusion matrix



XGBoost: code chunk and metrics

```r
library(xgboost)

# 0. splitting the dataset into training and test sets
set.seed(2023)
ind <- sample(2, nrow(iris),replace=TRUE,prob=c(0.7,0.3))
training <- iris[ind==1,]
testing <- iris[ind==2,]

xgb_training = xgb.DMatrix(data = as.matrix(training[,-5]), label = training
    [,5])
xgb_testing = xgb.DMatrix(data = as.matrix(testing[,-5]), label = testing[,5])

# 1. Build a XGBoost Classifier
set.seed(2023)
xgb_model <- xgboost(data=xgb_training, max.depth=3, nrounds=50)

pred_xgb_testing <- predict(xgb_model, xgb_testing)
pred_y_xgb_testing = as.factor((levels(testing[,5]))[round(pred_xgb_testing)])
```

	Training	Testing
accuracy	100.00	100.00
precision	100.00	100.00
recall	100.00	100.00
f1-score	100.00	100.00

References

The R Project for Statistical Computing:

https://www.r-project.org/

https://www.v7labs.com/blog/confusion-matrix-guide