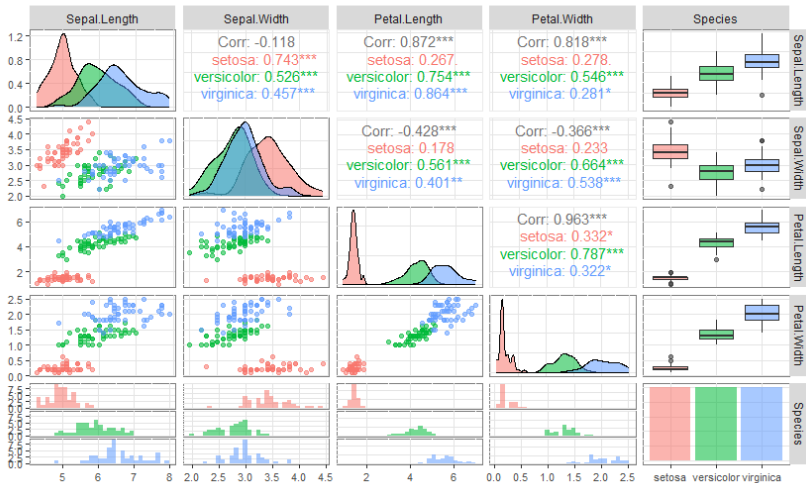


# Visualization of Iris dataset

## Summary of distributions

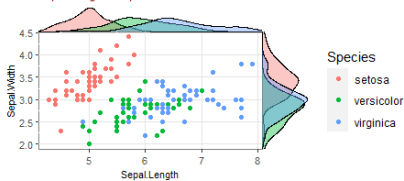
Complete Iris dataset



# Scatterplots with marginal densities

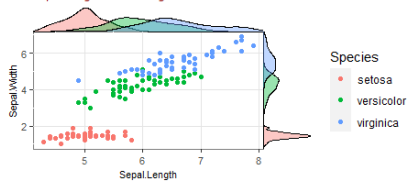
Scatterplot with marginal densities

*Sepal.Length x Sepal.Width from Iris dataset*



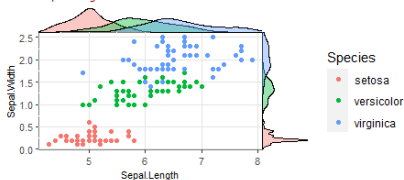
Scatterplot with marginal densities

*Sepal.Length x Petal.Length from Iris dataset*



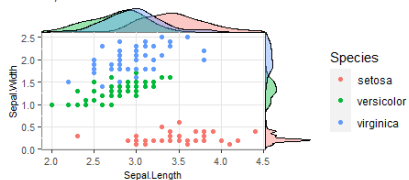
Scatterplot with marginal densities

*Sepal.Length x Petal.Width from Iris dataset*



Scatterplot with marginal densities

*Sepal.Width x Petal.Length from 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.

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

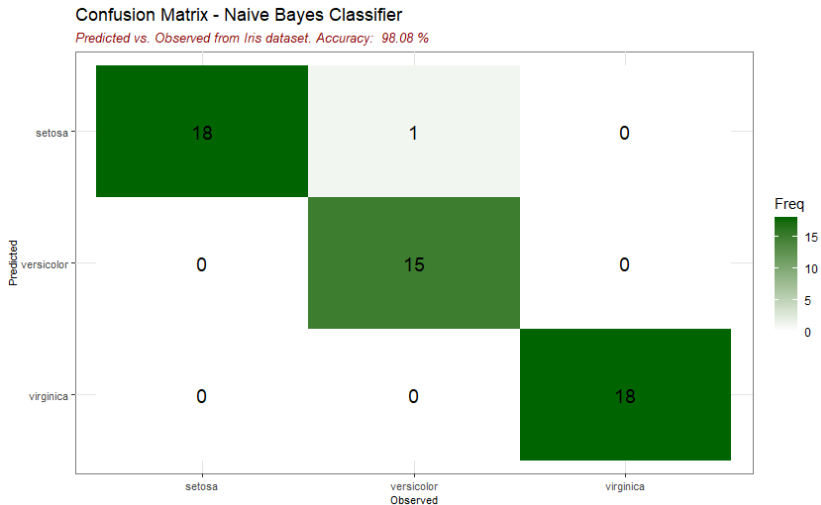
Recall (for the positive class). The number of samples predicted correctly to be belonging to the positive class out of all the samples that actually belong to the positive class.

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

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

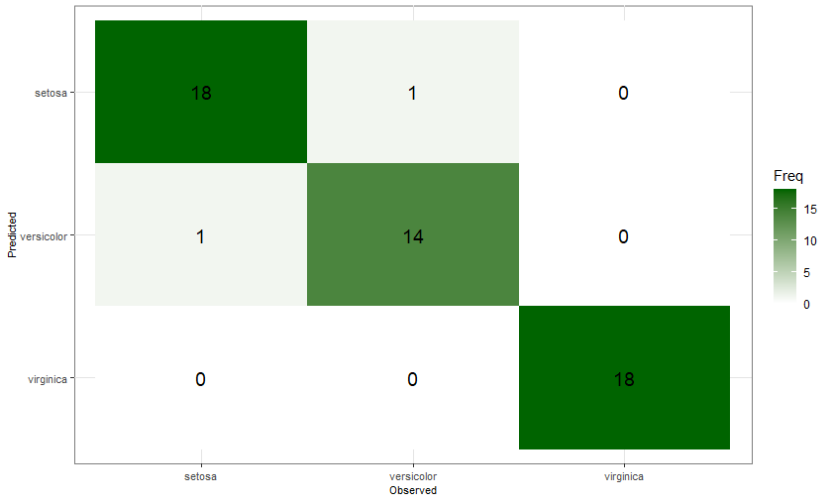
```
1 library(klaR)
2
3 # 1. Build a Naive Bayes Classifier
4 set.seed(2023)
5 nb_model <- NaiveBayes(Species ~ ., data=training) # train Na ve Bayes model
6 pred_nb <- predict(nb_model, testing) # apply Na ve Bayes model on test set
7 pred_nb_training <- predict(nb_model, training) # apply Na ve Bayes 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

Confusion Matrix - Random Forest Classifier

*Predicted vs. Observed from Iris testing dataset. Accuracy: 96.15 %*



# Random Forest: code chunk and metrics

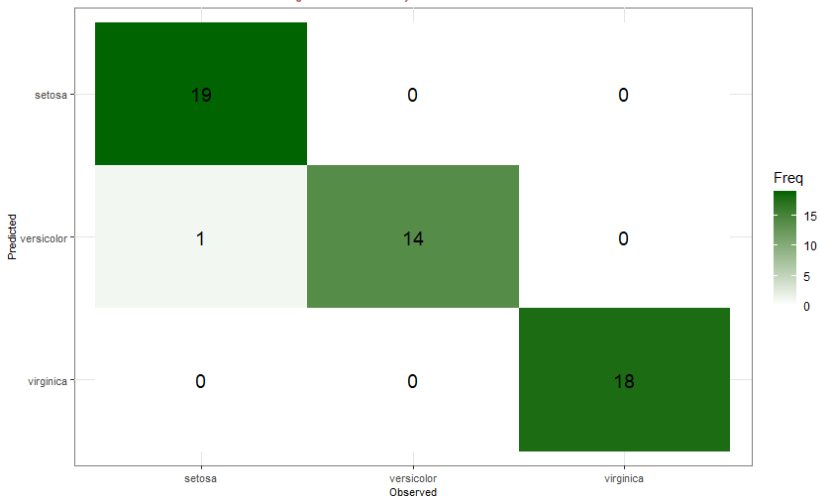
```
1 library(randomForest)
2
3 # 1. Build a Random Forest learning tree Classifier
4 set.seed(2023)
5 rf_model <- randomForest(Species~., data=training, ntree=100, proximity=TRUE) #
  train RF model
6 pred_rf <- predict(rf_model, testing) # apply RF model on test set
7 pred_rf_training <- predict(rf_model, 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

Confusion Matrix - Multinomial Logistic Regression Classifier

*Predicted vs. Observed from Iris testing dataset. Accuracy: 98.08 %*





# Logistic Regression: code chunk and metrics

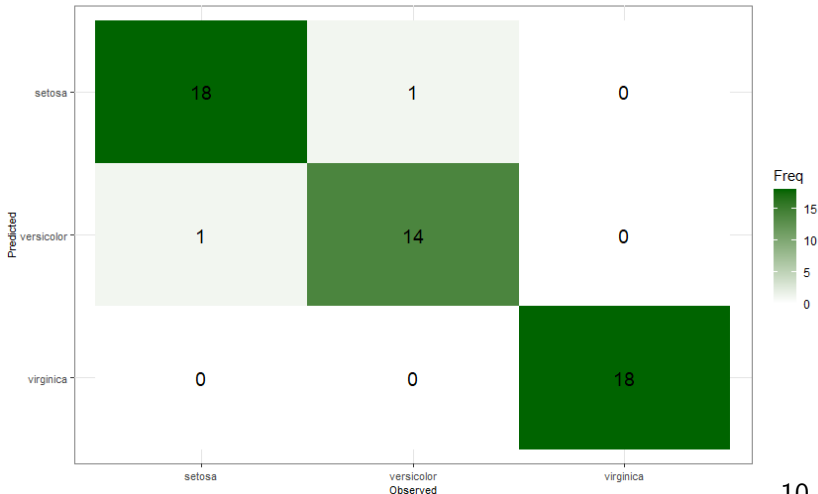
```
1 library(stats4)
2 library(splines)
3 library(VGAM)
4
5 # 1. Build a Multinomial Logistic Regression Classifier
6 set.seed(2023)
7 mlr_model <- vglm(Species ~ ., family=multinomial, training)
8 pred_mlr_training<- predict(mlr_model, training, type = "response")
9 pred_mlr_testing<- predict(mlr_model, testing, type = "response")
10 predictions <- apply(pred_mlr_testing, 1, which.max)
11 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

Confusion Matrix - Support Vector Machines Classifier

*Predicted vs. Observed from Iris testing dataset. Accuracy: 96.15 %*



# Support Vector Machines: code chunk and metrics

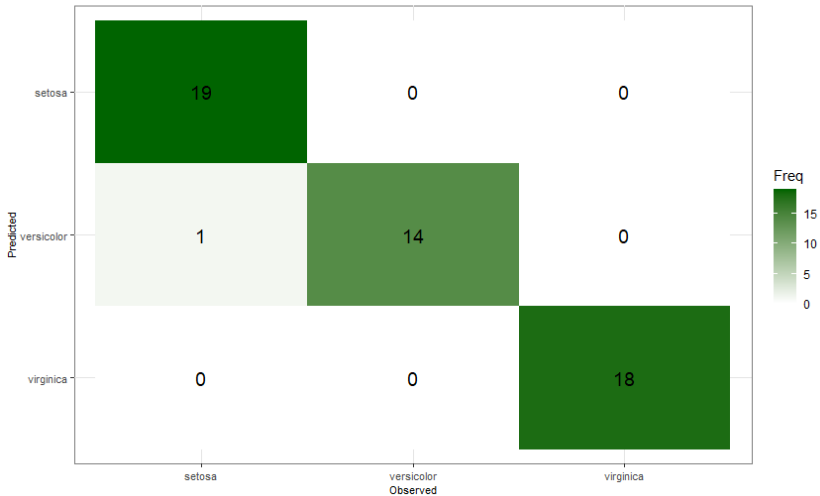
```
1 library(e1071)
2
3 # 1. Build a Support Vector Machines Classifier
4 set.seed(2023)
5 svm_model <- svm(Species ~ ., data=training,
6                  kernel="radial") #linear/polynomial/sigmoid
7 pred_svm <- predict(svm_model, testing)
8 pred_svm_training <- predict(svm_model, training) # apply 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

Confusion Matrix - Neural Networks Classifier

*Predicted vs. Observed from Iris testing dataset. Accuracy: 98.08 %*



# Neural Networks: code chunk and metrics

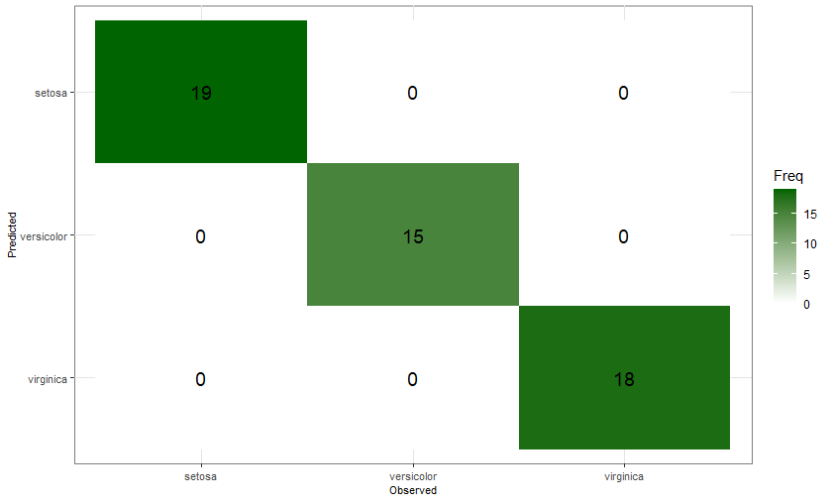
```
1 library(neuralnet)
2
3 # 1. Build a Neural Network Classifier
4 set.seed(2023)
5 iris$setosa <- iris$Species=="setosa"
6 iris$virginica <- iris$Species == "virginica"
7 iris$versicolor <- iris$Species == "versicolor"
8
9 # splitting into test and training again because we added variables
10 ind <- sample(2, nrow(iris), replace=TRUE, prob=c(0.7,0.3))
11 training <- iris[ind==1,] ; testing <- iris[ind==2,]
12
13 nn_model <- neuralnet(setosa+versicolor+virginica ~
14     Sepal.Length + Sepal.Width + Petal.Length + Petal.Width,
15     data=training, hidden=c(10,10), rep = 5, err.fct = "ce",
16     linear.output = F, lifestign = "minimal",
17     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

Confusion Matrix - Decision Tree Classifier

*Predicted vs. Observed from Iris testing dataset. Accuracy: 100 %*



# Decision Tree: code chunk and metrics

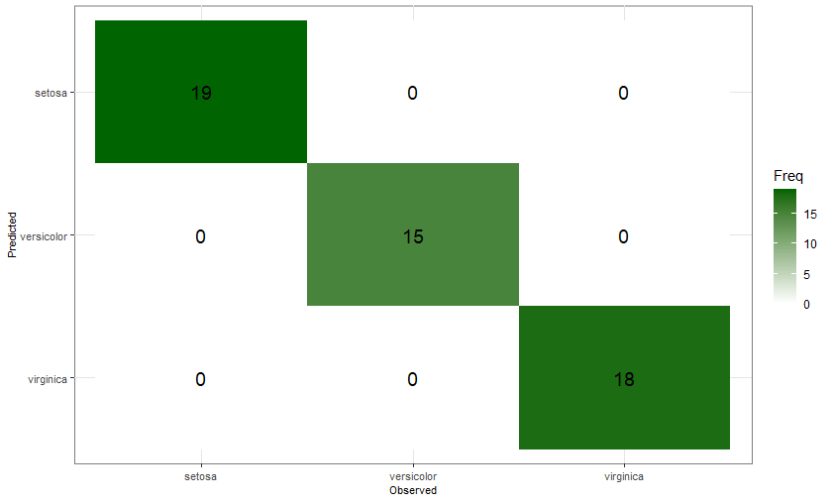
```
1 library(rpart)
2
3 # 1. Build a Decision Tree Classifier
4 set.seed(2023)
5 dt_model <- rpart(Species ~.,
6                   data = training,
7                   method = "class",
8                   control = rpart.control(cp = 0),
9                   parms = list(split = "information"))
10
11 pred_dt_training<- predict(dt_model, training, type = "class")
12 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

Confusion Matrix - XGBoost Classifier

*Predicted vs. Observed from Iris testing dataset. Accuracy: 100 %*



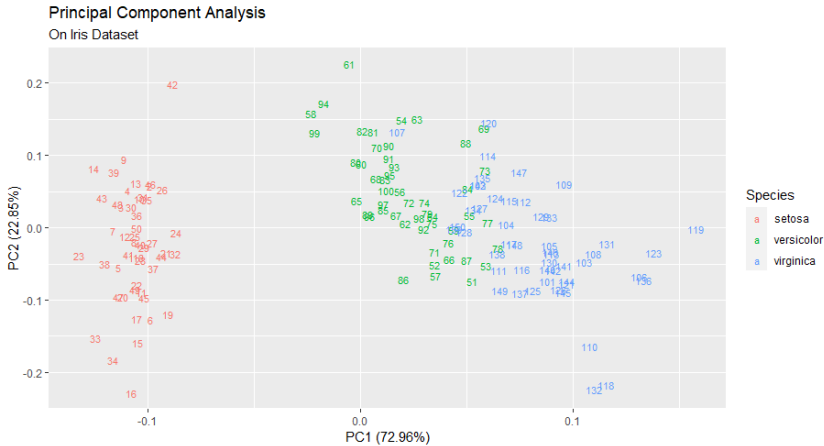


# XGBoost: code chunk and metrics

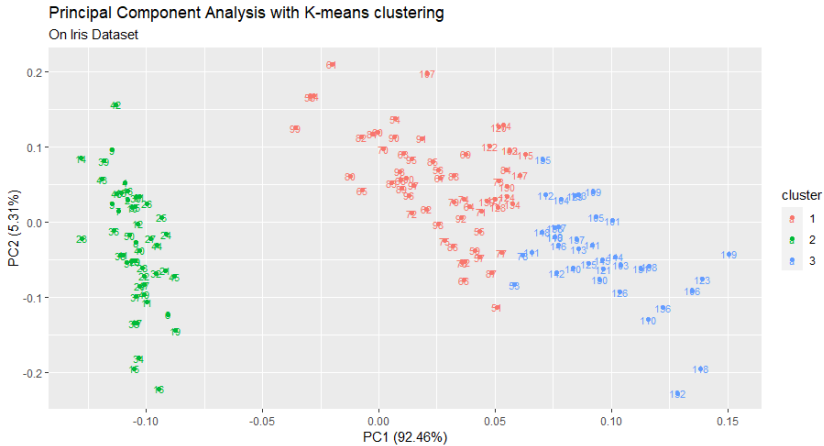
```
1 library(xgboost)
2
3 # 0. splitting the dataset into training and test sets
4 set.seed(2023)
5 ind <- sample(2, nrow(iris), replace=TRUE, prob=c(0.7,0.3))
6 training <- iris[ind==1,]
7 testing <- iris[ind==2,]
8
9 xgb_training = xgb.DMatrix(data = as.matrix(training[,-5]), label = training
10                             [,5])
11 xgb_testing = xgb.DMatrix(data = as.matrix(testing[,-5]), label = testing[,5])
12
13 # 1. Build a XGBoost Classifier
14 set.seed(2023)
15 xgb_model <- xgboost(data=xgb_training, max.depth=3, nrounds=50)
16
17 pred_xgb_testing <- predict(xgb_model, xgb_testing)
18 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  |

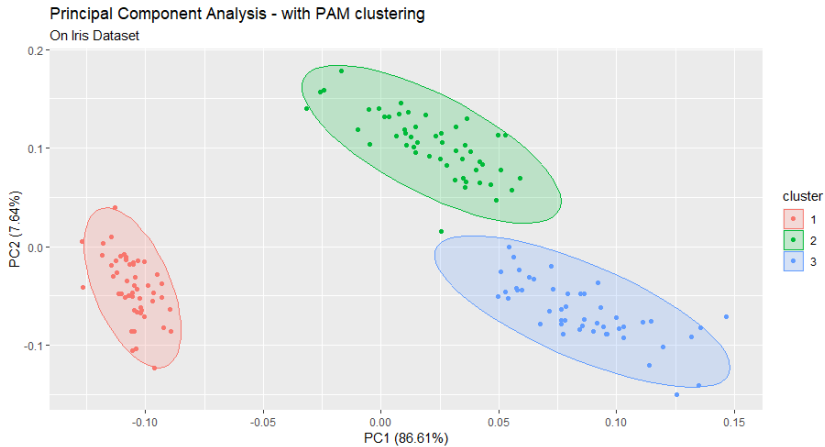
# Principal Component Analysis



# K-means



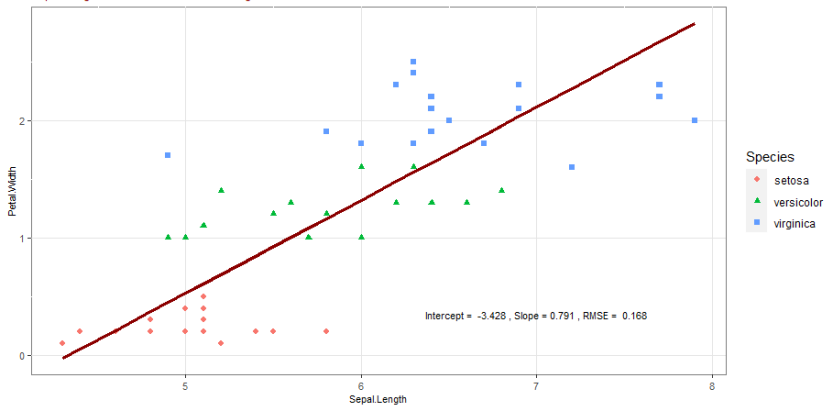
# Partitioning Around Medoids



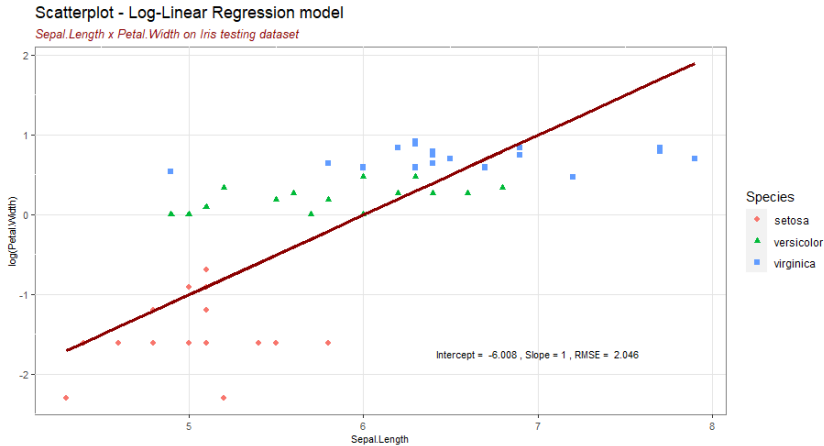
# Simple Linear Regression

Scatterplot - Linear Regression model

*Sepal.Length x Petal.Width on Iris testing dataset*



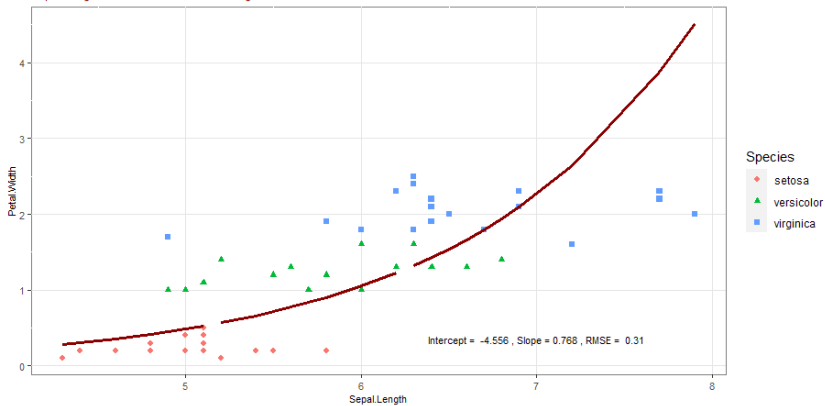
# Simple Log-Linear Regression



# Poisson Regression

## Scatterplot - Poisson Regression model

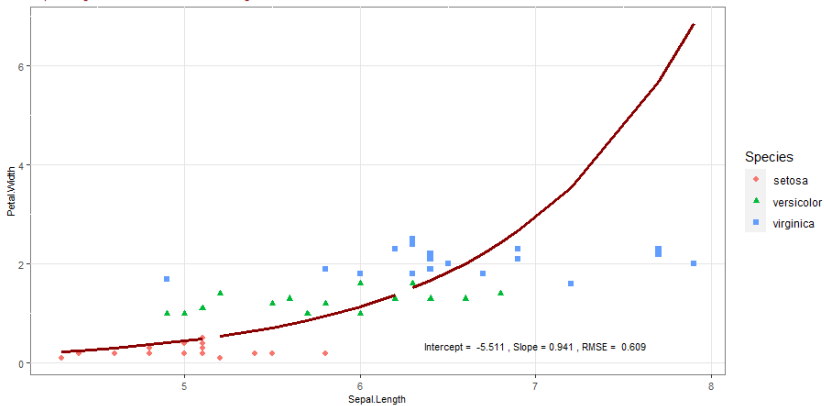
*Sepal.Length x Petal.Width on Iris testing dataset*



# Gamma Regression

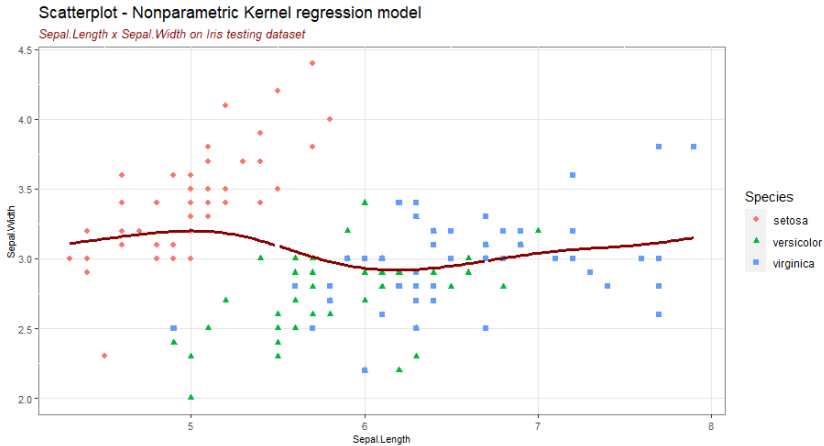
Scatterplot - Gamma Regression model

*Sepal.Length x Petal.Width on Iris testing dataset*





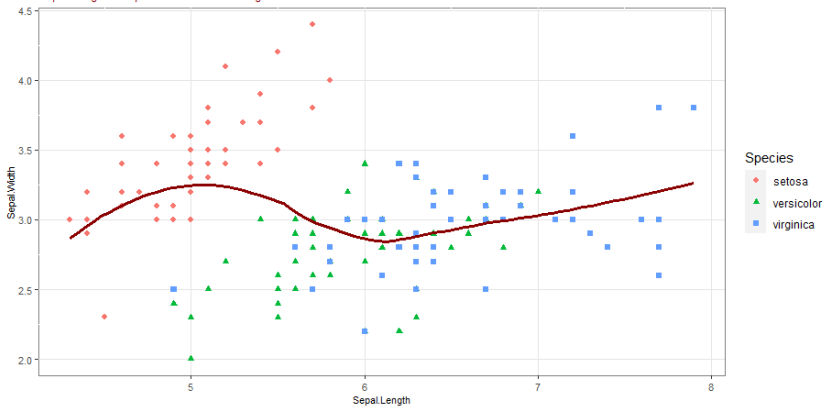
# Nonparametric Kernel Regression



# Nonparametric Smoothing Splines Regression

Scatterplot - Nonparametric Smoothing Splines regression model

*Sepal.Length x Sepal.Width on Iris testing dataset*



# References

The R Project for Statistical Computing:

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

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