# IRIS FLOWER CLASSIFICATION

*Project Work by Sreya Bhattacharya*

SUBMITTED FOR DATA SCIENCE TASK IN CODSOFT FOR INTERNSHIP

## Introduction:

This project is an application of machine learning with R programming. The dataset consists 150 data of which 50 observations are on Iris Setosa, 50 observations on Iris Versicolor and the rest 50 of Iris Virginica. We have the data based on sepal length, sepal width, and petal length & petal width. We'll fit a model in this dataset that can classify iris flowers into different species based on their sepal and petal measurements.

## Data Source:

I've collected the dataset from the given link in the task.

(https://www.kaggle.com/datasets/arshid/iris-flower-dataset)

## Calculations and Analysis:

I have used R-Studio for all calculations.

- First, I've loaded the data at R Studio. That is-

| | sepal_length | sepal_width | petal_length | petal_width | species |
|---|---|---|---|---|---|
| 1 | 5.1 | 3.5 | 1.4 | 0.2 | Iris-setosa |
| 2 | 4.9 | 3.0 | 1.4 | 0.2 | Iris-setosa |
| 3 | 4.7 | 3.2 | 1.3 | 0.2 | Iris-setosa |
| 4 | 4.6 | 3.1 | 1.5 | 0.2 | Iris-setosa |
| 5 | 5.0 | 3.6 | 1.4 | 0.2 | Iris-setosa |
| 6 | 5.4 | 3.9 | 1.7 | 0.4 | Iris-setosa |
| 7 | 4.6 | 3.4 | 1.4 | 0.3 | Iris-setosa |
| 8 | 5.0 | 3.4 | 1.5 | 0.2 | Iris-setosa |
| 9 | 4.4 | 2.9 | 1.4 | 0.2 | Iris-setosa |
| 10 | 4.9 | 3.1 | 1.5 | 0.1 | Iris-setosa |

Then the calculations and the analysis are done as follows.

- Next we'll see the internal structure of the data.

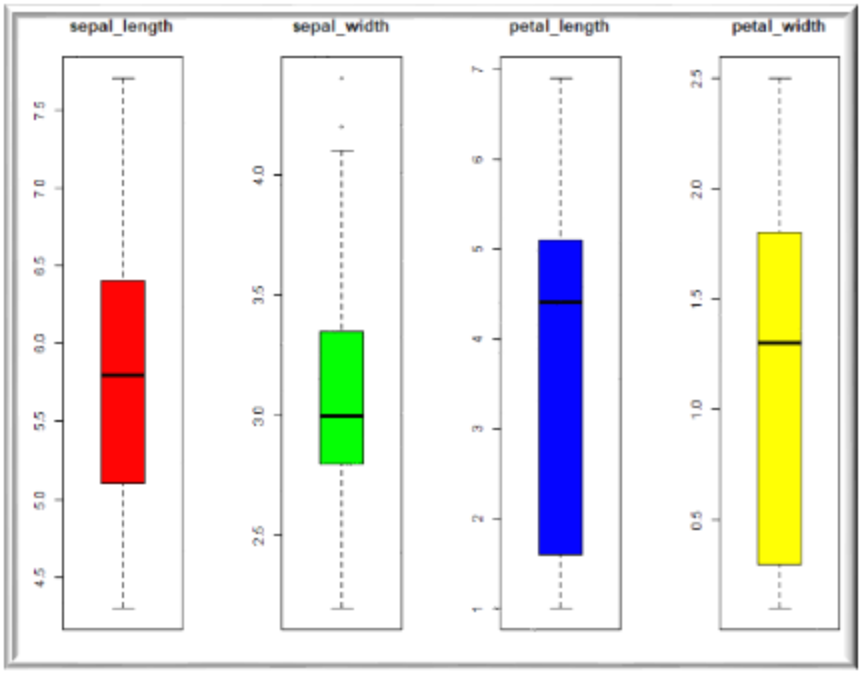
```
spc_tbl_ [150 × 5] (S3: spec_tbl_df/tbl_df/tbl/data.frame)
 $ sepal_length: num [1:150] 5.1 4.9 4.7 4.6 5 5.4 4.6 5 4.4 4.9 ...
 $ sepal_width : num [1:150] 3.5 3 3.2 3.1 3.6 3.9 3.4 3.4 2.9 3.1 ...
 $ petal_length: num [1:150] 1.4 1.4 1.3 1.5 1.4 1.7 1.4 1.5 1.4 1.5 ...
 $ petal_width : num [1:150] 0.2 0.2 0.2 0.2 0.2 0.4 0.3 0.2 0.2 0.1 ...
 $ species     : chr [1:150] "Iris-setosa" "Iris-setosa" "Iris-setosa" "Iris-setosa" ...
 - attr(*, "spec")=
  .. cols(
  ..   sepal_length = col_double(),
  ..   sepal_width = col_double(),
  ..   petal_length = col_double(),
  ..   petal_width = col_double(),
  ..   species = col_character()
  .. )
 - attr(*, "problems")=<externalptr>
```

The dataset "IRIS" contains information of 5 variables for 150 observations. The first 4 columns, Sepal Length, Sepal Width, Petal Length, and Petal Width, contain numeric values but the last one contains characters.

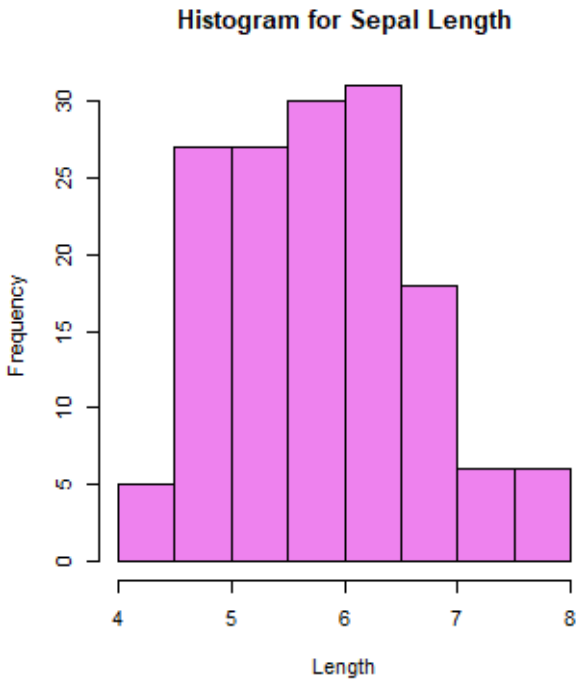
- Summary of the data is as follows,

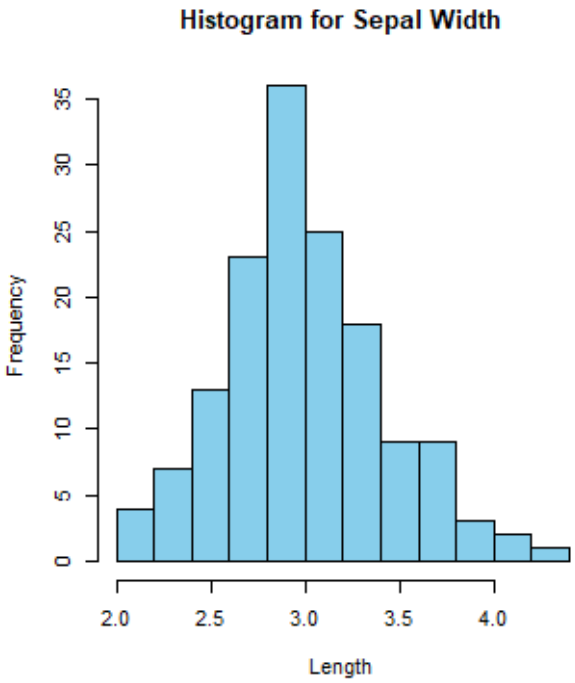
```
> summary(IRIS)
  sepal_length    sepal_width     petal_length    petal_width          species
 Min.   :4.300   Min.   :2.000   Min.   :1.000   Min.   :0.100   Length:150
 1st Qu.:5.100   1st Qu.:2.800   1st Qu.:1.600   1st Qu.:0.300   Class :character
 Median :5.800   Median :3.000   Median :4.350   Median :1.300   Mode  :character
 Mean   :5.843   Mean   :3.054   Mean   :3.759   Mean   :1.199
 3rd Qu.:6.400   3rd Qu.:3.300   3rd Qu.:5.100   3rd Qu.:1.800
 Max.   :7.900   Max.   :4.400   Max.   :6.900   Max.   :2.500
```

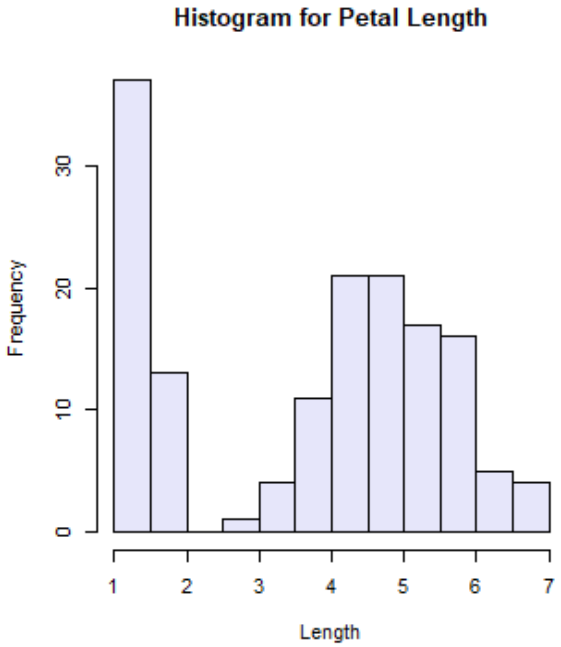
- Next we'll see boxplot of the 4 independent variables.

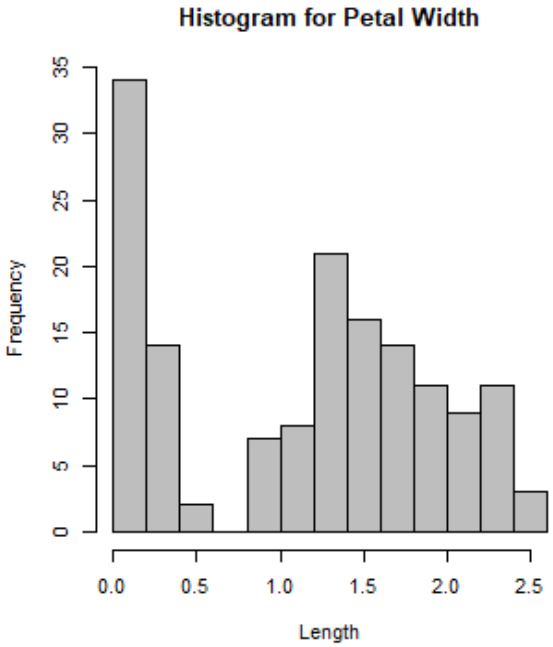


- Now we'll see the histograms of petal length, sepal length, sepal width and petal width.



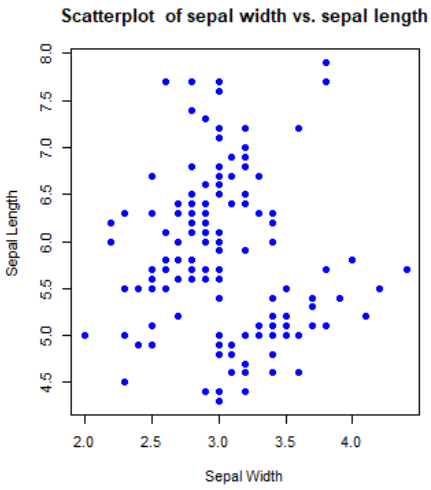



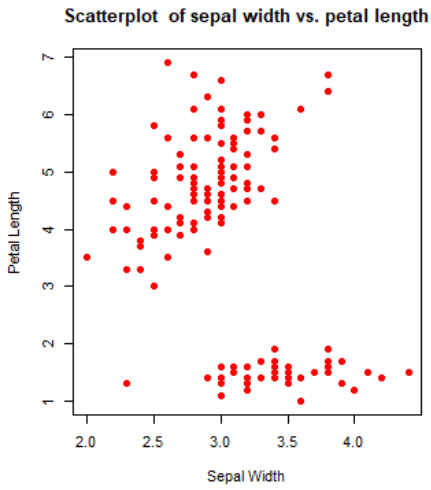



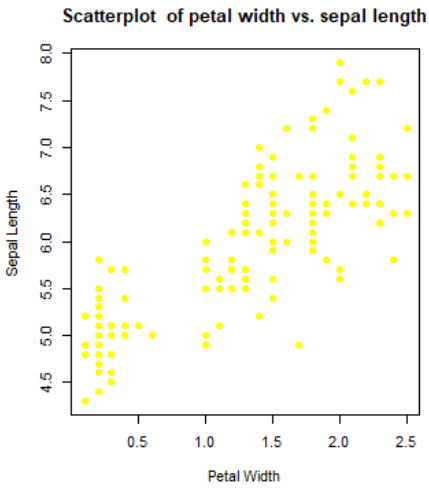

- Now we'll see the percentage of data we have of each class of Iris flowers.

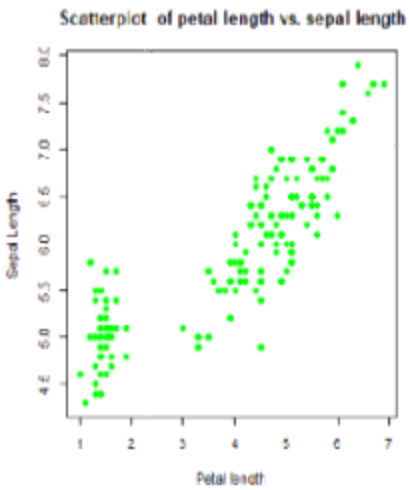
| | freq | percentage |
|---|---|---|
| Iris-setosa | 50 | 33.33333 |
| Iris-versicolor | 50 | 33.33333 |
| Iris-virginica | 50 | 33.33333 |

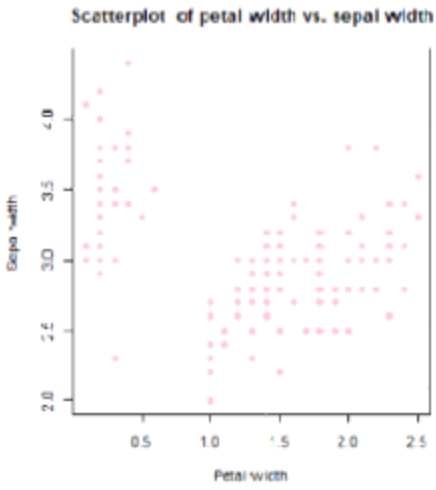
- Next we'll see the scatter plots of the variables.

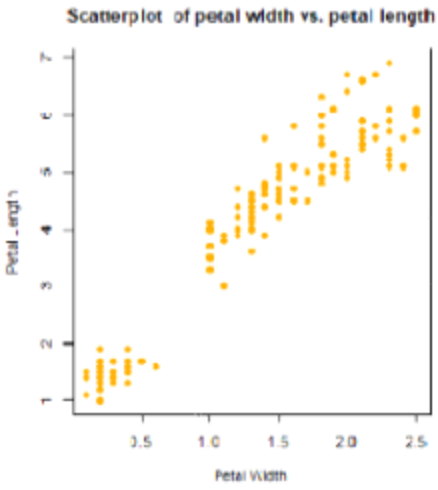


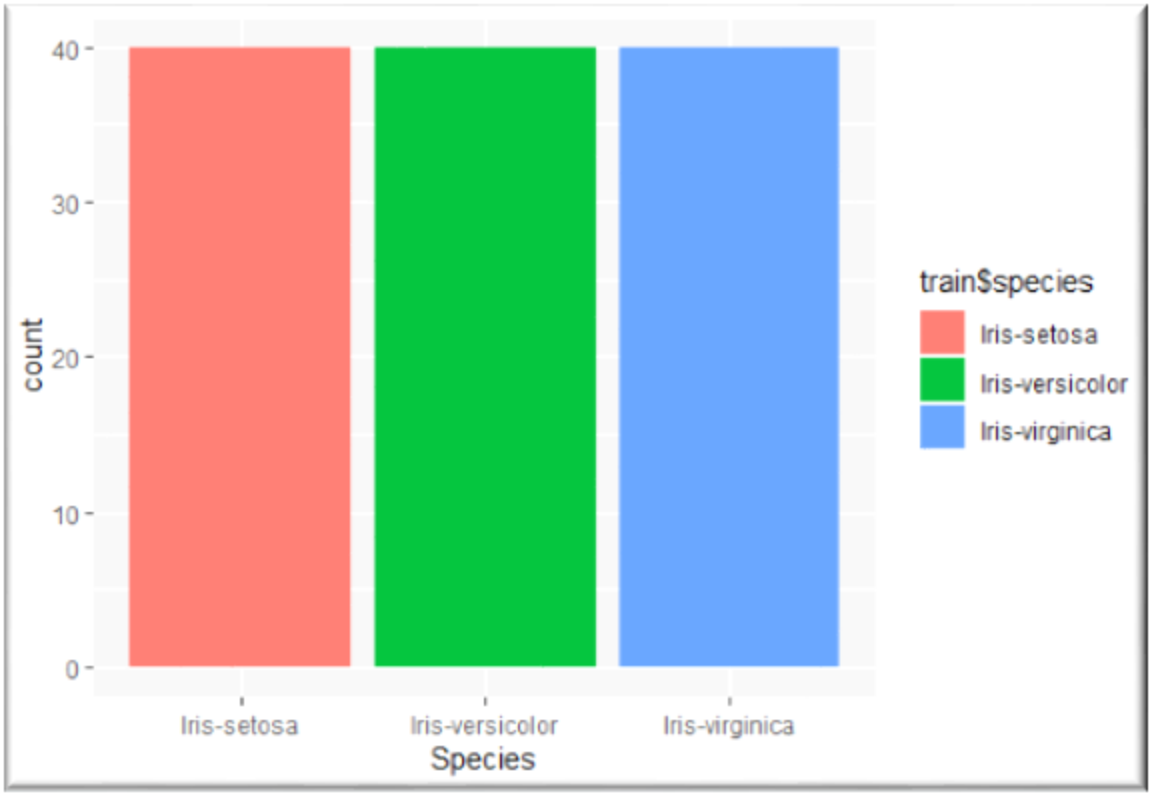




- After splitting the dataset into Train and Test let us see the dimension of the train and test data.

```
> dim(train)
[1] 120   5
> dim(test)
[1] 30  5
```

- Count of three types of flowers in Train Dataset is as follows,

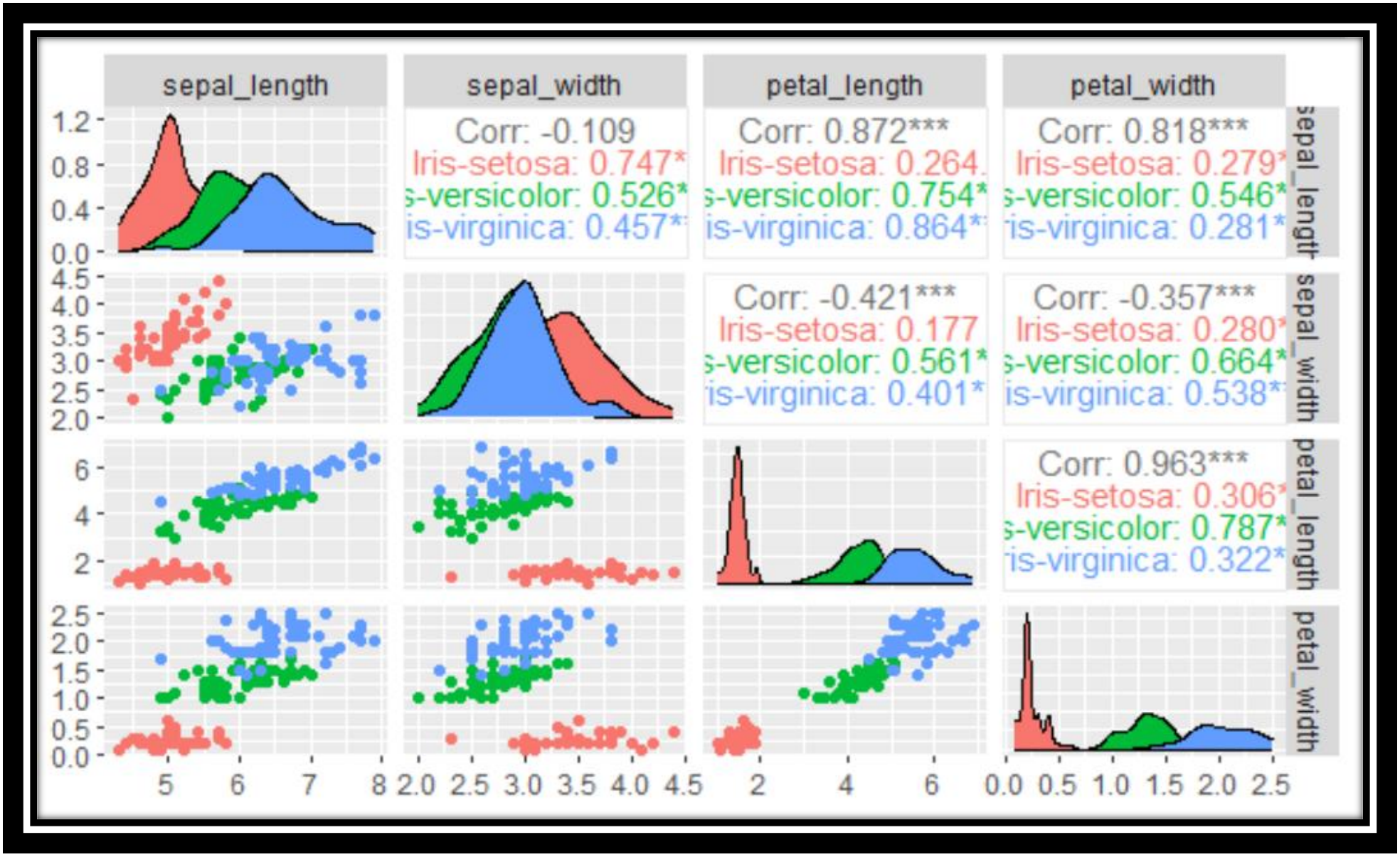



So, we can see in our Train dataset there are 40 numbers of Setosa, 40 numbers of Versicolor and 40 numbers of Virginica.

- Next we'll see the average values of the variables for three types of Iris Flowers.

| | species | avg_SL | avg_SW | avg_PL | avg_PW |
|---|---|---|---|---|---|
| | <chr> | <dbl> | <dbl> | <dbl> | <dbl> |
| 1 | Iris-setosa | 5.00 | 3.42 | 1.46 | 0.23 |
| 2 | Iris-versicolor | 6.00 | 2.81 | 4.28 | 1.34 |
| 3 | Iris-virginica | 6.58 | 2.98 | 5.58 | 2.03 |

- Next we'll create a scatter plot matrix.


sepal_length
sepal_width
petal_length
petal_width
Corr: -0.109
Iris-setosa: 0.747*
s-versicolor: 0.526*
is-virginica: 0.457*
Corr: 0.872***
Iris-setosa: 0.264
s-versicolor: 0.754*
is-virginica: 0.864*
Corr: 0.818***
Iris-setosa: 0.279*
s-versicolor: 0.546*
is-virginica: 0.281*
Corr: -0.421***
Iris-setosa: 0.177
s-versicolor: 0.561*
is-virginica: 0.401*
Corr: -0.357***
Iris-setosa: 0.280*
s-versicolor: 0.664*
is-virginica: 0.538*
Corr: 0.963***
Iris-setosa: 0.306*
s-versicolor: 0.787*
is-virginica: 0.322*
sepal_length
sepal_width
petal_length
petal_width

- Next we'll fit a LDA (Latent Dirichlet Allocation) model in this data. After fitting the model we'll see the accuracy and the value of kappa coefficient.i.e.-

```
Linear Discriminant Analysis

120 samples
  4 predictor
  3 classes: 'Iris-setosa', 'Iris-versicolor', 'Iris-virginica'

No pre-processing
Resampling: Cross-Validated (10 fold)
Summary of sample sizes: 108, 108, 108, 108, 108, 108, ...
Resampling results:

  Accuracy   Kappa
  0.9833333  0.975
```

From the accuracy and the kappa value we can say that we chose the right model.

- Now we'll fit the model in the test data and see the accuracy.
  - The Confusion Matrix is as follows –

```
> confusion_matrix

lda_predict       Iris-setosa Iris-versicolor Iris-virginica
  Iris-setosa              10               0              0
  Iris-versicolor           0               9              0
  Iris-virginica            0               1             10
```

So we can see from the confusion matrix our most of the predictions are correct in the test data series.

- The accuracy is as follows,

```
> print(accuracy)
[1] 0.9666667
```

## Conclusion:

I've divided the dataset in two parts "Train" & "Test". The Train dataset contains 120 observations and the Test dataset contains 30 observations.

Here we fitted a LDA (Latent Dirichlet Allocation) model in our train dataset and based on the accuracy and kappa coefficient value we can consider that the model we've chosen fits our data very well.

Next we fitted the data in our Train dataset and from the calculations we can see that our prediction is 96.6667% accurate.

## Reference:


1. Fundamentals of Machine Learning for Predictive Data Analytics by John D. Kelleher, Brian Mac Namee, and Aoife D'Arcy.
2. Programming Collective Intelligence by Toby Segaran
3. Hands-On Programming with R by Garrett Grolemund & Hadley Wickham.
4. R in action by Dr. Robert L. Kabacoff.

## Acknowledgement:

I would like to express my special thanks of gratitude to CODSOFT for giving me the golden opportunity to do the project on the wonderful topic which helped me doing a lot of research and I came to know about so many things.

It helped me increase my knowledge and skills.

**Annexure:**

```
library(readr)
IRIS <- read_csv("C:\\Users\\USER\\Downloads\\IRIS.csv")
View(IRIS)
str(IRIS)
summary(IRIS)
x <- IRIS[ , 1:4]
y <- IRIS[ , 5]
levels(IRIS$species)
par(mfrow=c(1,4))
color <- c("red", "green", "blue", "yellow")
for (i in 1:4) {
  boxplot(train[, -c(5)][i], main=names(train)[i], col = color[i] )
}
hist(IRIS$sepal_length,
     col='violet',
     main='Histogram for Sepal Length',
     xlab='Length',
     ylab='Frequency')
hist(IRIS$sepal_width,
     col='skyblue',
     main='Histogram for Sepal Width',
     xlab='Length',
     ylab='Frequency')
hist(IRIS$petal_length,
     col='lavender',
     main='Histogram for Petal Length',
     xlab='Length',
```

```
    ylab='Frequency')
hist(IRIS$petal_width,
    col='grey',
    main='Histogram for Petal Width',
    xlab='Length',
    ylab='Frequency')
percentage <- prop.table(table(IRIS$species)) * 100
cbind(freq=table(IRIS$species), percentage=percentage)
plot(IRIS$sepal_width, IRIS$sepal_length,
    col='blue',
    main='Scatterplot of sepal width vs. sepal length',
    xlab='Sepal Width',
    ylab='Sepal Length',
    pch=19)
plot(IRIS$sepal_width, IRIS$petal_length,
    col='red',
    main='Scatterplot of sepal width vs. petal length',
    xlab='Sepal Width',
    ylab='Petal Length',
    pch=19)
plot(IRIS$petal_width, IRIS$sepal_length,
    col='yellow',
    main='Scatterplot of petal width vs. sepal length',
    xlab='Petal Width',
    ylab='Sepal Length',
    pch=19)
plot(IRIS$petal_length, IRIS$sepal_length,
    col='green',
    main='Scatterplot of petal length vs. sepal length',
```

```
    xlab='Petal length',
    ylab='Sepal Length',
    pch=19)
plot(IRIS$petal_width, IRIS$sepal_width,
    col='pink',
    main='Scatterplot  of petal width vs. sepal width',
    xlab='Petal width',
    ylab='Sepal width',
    pch=19)
plot(IRIS$petal_width, IRIS$petal_length,
    col='orange',
    main='Scatterplot  of petal width vs. petal length',
    xlab='Petal Width',
    ylab='Petal Length',
    pch=19)
set.seed(100)
library(caret)
split <- createDataPartition(IRIS$species,  p = 0.8, list=FALSE)
train<-IRIS[split,]
test<-IRIS[-split,]
dim(train)
dim(test)
library(tidyverse)
library(dplyr)
count(train$species)
count(test$species)
qplot(x = train$species, fill = train$species, xlab = "Species", ylab ="count")
library(dplyr)
library(sf)
```

```
train %>%
  group_by(species) %>% summarise(avg_SL = mean(sepal_length), avg_SW = mean(sepal_width), avg_PL = mean(petal_length), avg_PW = mean(petal_width))
library(GGally)
# Create a scatter plot matrix
ggpairs(IRIS[, c( "sepal_length", "sepal_width", "petal_length", "petal_width")],
        aes(color = IRIS$species))
correlation <-cor(train[,c(1:4)], method = 'pearson')
control <- trainControl(method='cv', number=10)
metric <- 'Accuracy'
set.seed(123)
lda_fit <- train(species~., data=train, method='lda',
                 trControl=control, metric=metric)
lda_fit
lda_predict <- predict(lda_fit, test)
confusion_matrix<-table(lda_predict,test$species)
confusion_matrix
accuracy<-sum(diag(confusion_matrix))/sum(confusion_matrix)
print(accuracy)
```