Iris Classification

Rounak Saha

2024-10-07

About Project:

The Iris Classification project involves creating a machine learning model to classify iris flowers into three species (Setosa, Versicolour, and Virginica) based on the length and width of their petals and sepals. This is a classic problem in machine learning and is often used as an introductory example for classification algorithms.

The columns in this dataset are:

Id

SepalLengthCm

SepalWidthCm

PetalLengthCm

PetalWidthCm

Species

Problem Statement:

- The model should achieve a high level of accuracy in classifying iris species.
- The model's predictions should be consistent and reliable, as measured by cross-validation.
- The final report should provide clear and comprehensive documentation of the project, including all code, visualizations, and findings.

By achieving these objectives, the project will demonstrate the ability to apply machine learning techniques to a classic classification problem, providing insights into the characteristics of different iris species and the effectiveness of various algorithms for this task.

```
## First, we will install and load the required packages
install.packages("tidyverse", repos="https://cloud.r-project.org/")
install.packages("readr", repos="https://cloud.r-project.org/")
install.packages("dplyr", repos="https://cloud.r-project.org/")
install.packages("ggcorrplot", repos="https://cloud.r-project.org/")
install.packages("gridExtra", repos="https://cloud.r-project.org/")
install.packages("ggplot2", repos="https://cloud.r-project.org/")
install.packages("lubridate",repos="https://cloud.r-project.org/")
install.packages("ggthemes",repos="https://cloud.r-project.org/")
install.packages("class",repos="https://cloud.r-project.org/")
install.packages("knitr",repos="https://cloud.r-project.org/")
```

```r
install.packages("GGally", repos="https://cloud.r-project.org/")
install.packages("caTools", repos="https://cloud.r-project.org/")
install.packages("caret", repos="https://cloud.r-project.org/")
install.packages("lattice", repos="https://cloud.r-project.org/")
```

```r
## Loading the libraries
library(tidyverse)
```

```
## -- Attaching core tidyverse packages ------------------------ tidyverse 2.0.0 --
## v dplyr     1.1.4     v readr     2.1.5
## v forcats   1.0.0     v stringr   1.5.0
## v ggplot2   3.5.1     v tibble    3.2.1
## v lubridate 1.9.3     v tidyr     1.3.1
## v purrr     1.0.2
## -- Conflicts ------------------------------------------ tidyverse_conflicts() --
## x dplyr::filter() masks stats::filter()
## x dplyr::lag()    masks stats::lag()
## i Use the conflicted package (<http://conflicted.r-lib.org/>) to force all conflicts to become errors
```

```r
library(readr)
library(dplyr)
library(ggcorrplot)
library(ggplot2)
library(lubridate)
library(ggthemes)
library(gridExtra)
```

```
##
## Attaching package: 'gridExtra'
##
## The following object is masked from 'package:dplyr':
##
##     combine
```

```r
library(class)
library(knitr)
```

```
## Warning: package 'knitr' was built under R version 4.3.3
```

```r
library(GGally)
```

```
## Registered S3 method overwritten by 'GGally':
##   method from
##   +.gg   ggplot2
```

```r
library(caTools)
```

```
## Warning: package 'caTools' was built under R version 4.3.3
```

```
library(caret)
```

```
## Loading required package: lattice
##
## Attaching package: 'caret'
##
## The following object is masked from 'package:purrr':
##
##     lift
```

```
library(lattice)
```

We will now load the dataframes using `read_csv()` function

```
## Import data into R Studio
iris <- read_csv("/Users/ronsmackbook/Desktop/Unified mentor/Iris.csv")
```

```
## Rows: 150 Columns: 5
## -- Column specification --------------------------------------------------------
## Delimiter: ","
## chr (1): Species
## dbl (4): SepalLengthCm, SepalWidthCm, PetalLengthCm, PetalWidthCm
##
## i Use `spec()` to retrieve the full column specification for this data.
## i Specify the column types or set `show_col_types = FALSE` to quiet this message.
```

```
## Let's begin exploring this dataset
## Basic Summarization functions in R
head(iris)
```

```
## # A tibble: 6 x 5
##   SepalLengthCm SepalWidthCm PetalLengthCm PetalWidthCm Species
##           <dbl>        <dbl>         <dbl>        <dbl> <chr>
## 1           5.1          3.5           1.4          0.2 Iris-setosa
## 2           4.9          3             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            3.6           1.4          0.2 Iris-setosa
## 6           5.4          3.9           1.7          0.4 Iris-setosa
```

```
tail(iris)
```

```
## # A tibble: 6 x 5
##   SepalLengthCm SepalWidthCm PetalLengthCm PetalWidthCm Species
##           <dbl>        <dbl>         <dbl>        <dbl> <chr>
## 1           6.7          3.3           5.7          2.5 Iris-virginica
## 2           6.7          3             5.2          2.3 Iris-virginica
## 3           6.3          2.5           5            1.9 Iris-virginica
## 4           6.5          3             5.2          2   Iris-virginica
## 5           6.2          3.4           5.4          2.3 Iris-virginica
## 6           5.9          3             5.1          1.8 Iris-virginica
```

```r
summary(iris)
```

```
##  SepalLengthCm    SepalWidthCm   PetalLengthCm    PetalWidthCm
##  Min.   :4.300   Min.   :2.000   Min.   :1.000   Min.   :0.100
##  1st Qu.:5.100   1st Qu.:2.800   1st Qu.:1.600   1st Qu.:0.300
##  Median :5.800   Median :3.000   Median :4.350   Median :1.300
##  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
##    Species
##  Length:150
##  Class :character
##  Mode  :character
##
##
##
```

```r
str(iris)
```

```
## spc_tbl_ [150 x 5] (S3: spec_tbl_df/tbl_df/tbl/data.frame)
##  $ SepalLengthCm: num [1:150] 5.1 4.9 4.7 4.6 5 5.4 4.6 5 4.4 4.9 ...
##  $ SepalWidthCm : num [1:150] 3.5 3 3.2 3.1 3.6 3.9 3.4 3.4 2.9 3.1 ...
##  $ PetalLengthCm: num [1:150] 1.4 1.4 1.3 1.5 1.4 1.7 1.4 1.5 1.4 1.5 ...
##  $ PetalWidthCm : 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(
##   ..   SepalLengthCm = col_double(),
##   ..   SepalWidthCm = col_double(),
##   ..   PetalLengthCm = col_double(),
##   ..   PetalWidthCm = col_double(),
##   ..   Species = col_character()
##   .. )
##  - attr(*, "problems")=<externalptr>
```

```r
dim(iris)
```

```
## [1] 150   5
```

The above functions provide a way for skimming through the dataset

`head()` - displays the first 6 entries

`tail()` - displays the last 6 entries

`summary()` - As stated, it summarizes the dataset

`dim()` - specifies the number of rows and columns in the dataset

`str()` - specifies the data type, variable names and the first few values

This is a better and simpler way to summarize mean of each columns grouped by species.

```
iris %>%
    group_by(Species)%>%
    summarize_if(is.numeric, mean)
```

```
## # A tibble: 3 x 5
##   Species         SepalLengthCm SepalWidthCm PetalLengthCm PetalWidthCm
##   <chr>                   <dbl>        <dbl>         <dbl>        <dbl>
## 1 Iris-setosa              5.01         3.42          1.46        0.244
## 2 Iris-versicolor          5.94         2.77          4.26        1.33
## 3 Iris-virginica           6.59         2.97          5.55        2.03
```

Findings

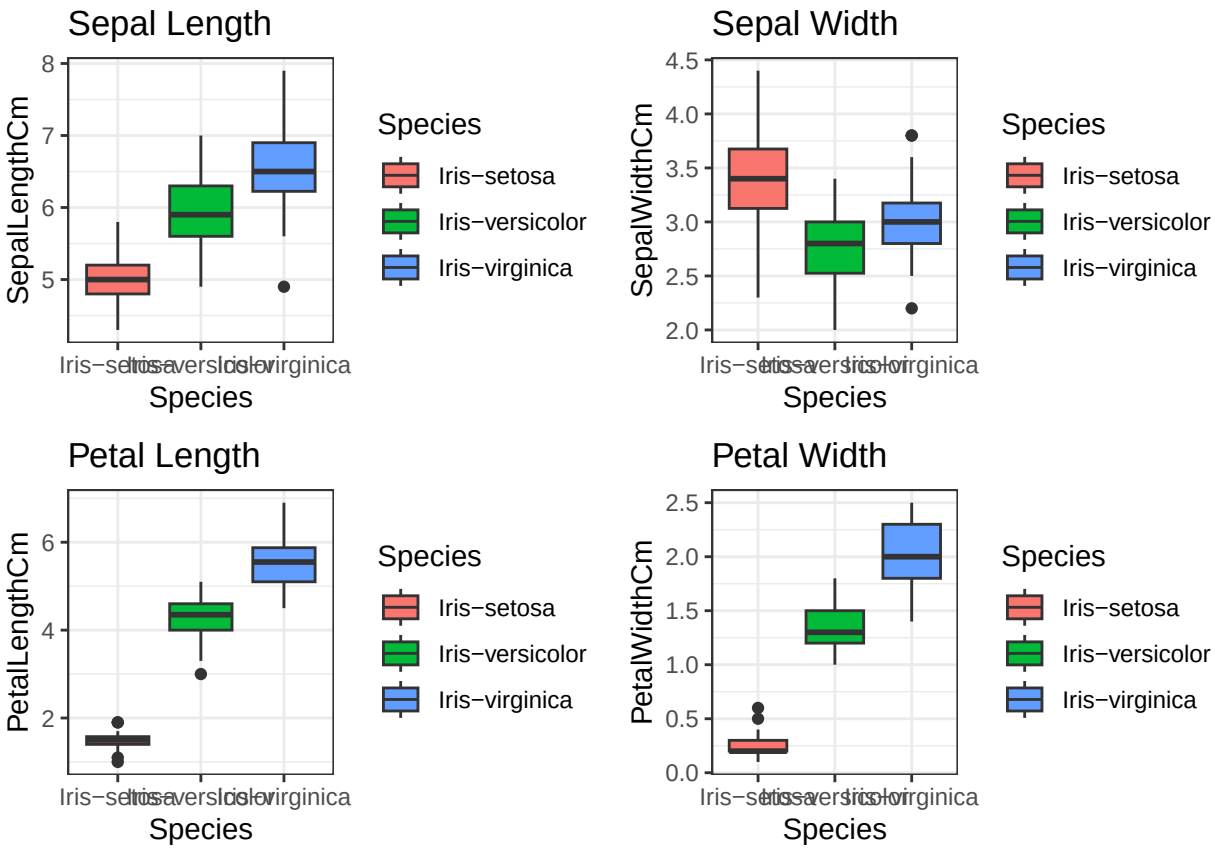
- Setosa species has the smallest petal length and petal width. Versicolor species has average petal length and petal width. Virginica species has the highest petal length and petal width.
- Versicolor species has the smallest sepal width. Virginica species came in second and setosa species has the largest sepal width.
- Virginica species has the longest sepal length, versicolor has the second longest sepal length and setosa species has the shortest sepal length.

Let's try and understand this dataset through a few visualizations. Before that we will edit the data a bit.

```
iris$Species <- as.factor(iris$Species)
```

Now, let us explore the variables by and look at the most convenient way to visualize these 4 variables? Box Plots of course!

```
## Code for box plot
p1 <- iris %>% ggplot(aes(x = Species, y = SepalLengthCm)) + theme_bw() +
  geom_boxplot(aes(fill = Species)) + labs(title = "Sepal Length", xlab = 'Species',
                                           ylab = 'Sepal Length (in cm))')

p2 <- iris %>% ggplot(aes(x = Species, y = SepalWidthCm)) + theme_bw() +
  geom_boxplot(aes(fill = Species)) + labs(title = "Sepal Width", xlab = 'Species',
                                           ylab = 'Sepal Width (in cm))')

p3 <- iris %>% ggplot(aes(x = Species, y = PetalLengthCm)) + theme_bw() +
  geom_boxplot(aes(fill = Species)) + labs(title = "Petal Length", xlab = 'Species',
                                           ylab = 'Petal Length (in cm))')

p4 <- iris %>% ggplot(aes(x = Species, y = PetalWidthCm)) + theme_bw() +
  geom_boxplot(aes(fill = Species)) + labs(title = "Petal Width", xlab = 'Species',
                                           ylab = 'Petal Width (in cm))')
grid.arrange(p1,p2,p3,p4, ncol =2)
```



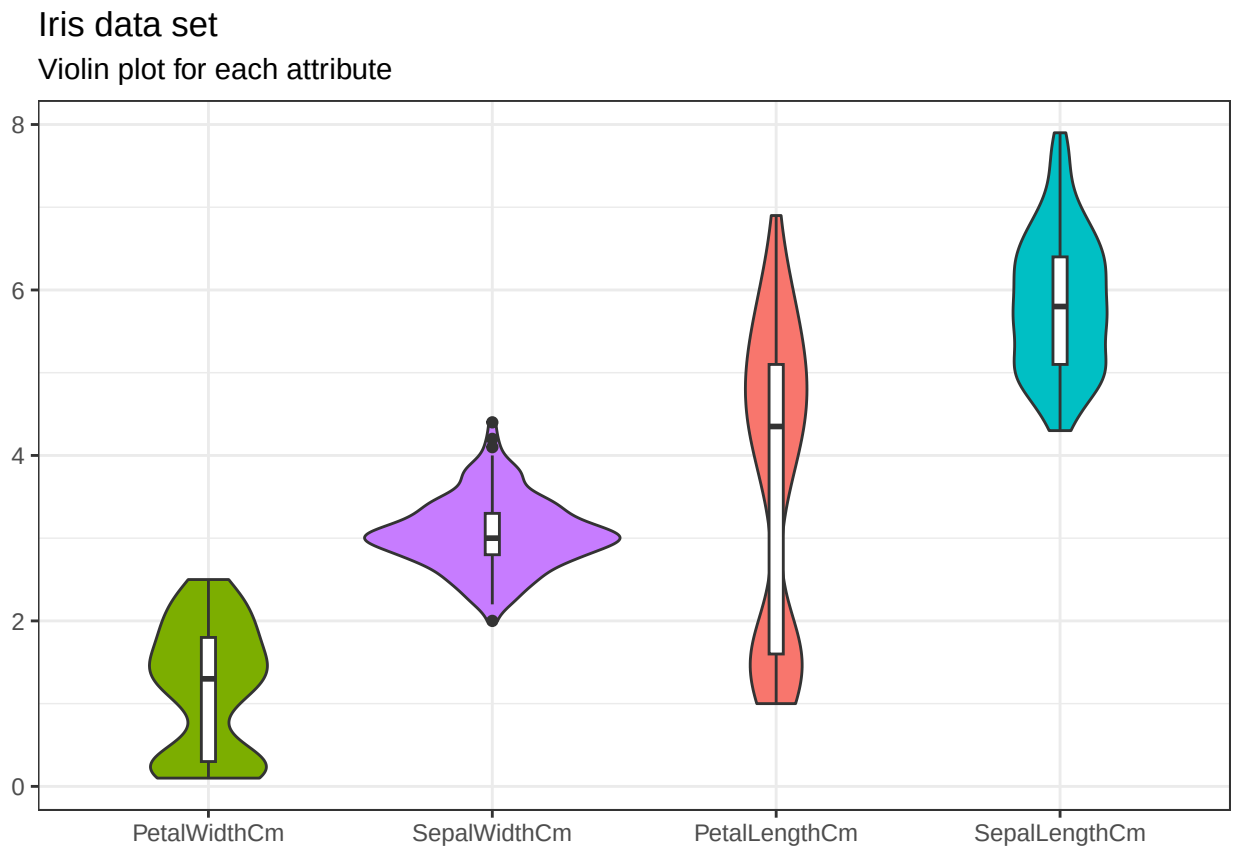
Box Plots are one very basic way to visualize the data. Another way is the violin plot and histogram. It's very similar to the box plot and it shows the distribution of points.

```
head(iris %>% gather(Attributes, value, 1:4))
```

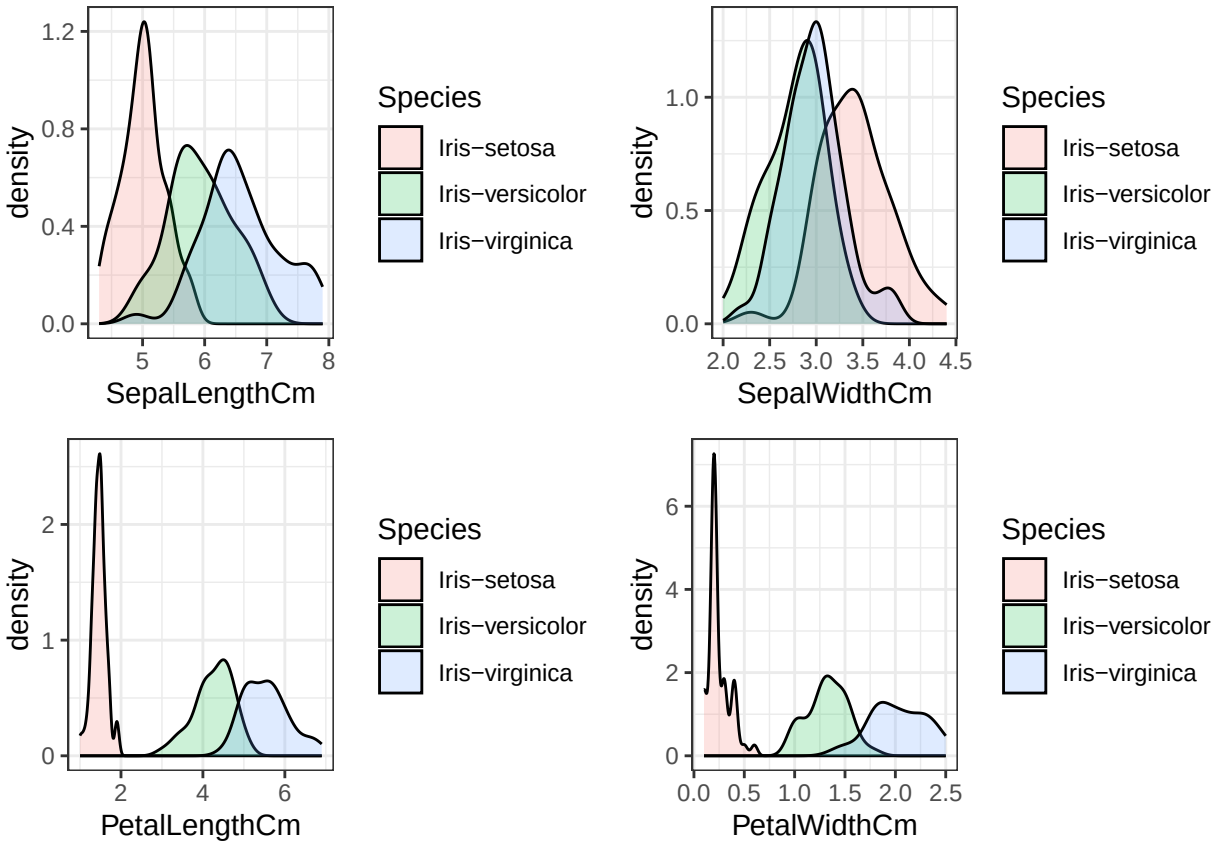
```
## # A tibble: 6 x 3
##   Species     Attributes    value
##   <fct>       <chr>         <dbl>
## 1 Iris-setosa SepalLengthCm   5.1
## 2 Iris-setosa SepalLengthCm   4.9
## 3 Iris-setosa SepalLengthCm   4.7
## 4 Iris-setosa SepalLengthCm   4.6
## 5 Iris-setosa SepalLengthCm   5
## 6 Iris-setosa SepalLengthCm   5.4
```

```
## Code for Violin Plot
p5 <- iris %>%
  gather(Attributes, value, 1:4) %>%
  ggplot(aes(x=reorder(Attributes, value, FUN=median), y=value, fill=Attributes)) +
  geom_violin(show.legend=FALSE) +
  geom_boxplot(width=0.05, fill="white") +
  labs(title="Iris data set",
       subtitle="Violin plot for each attribute") +
  theme_bw() +
  theme(axis.title.y=element_blank(),
```

```
        axis.title.x=element_blank())
p5
```

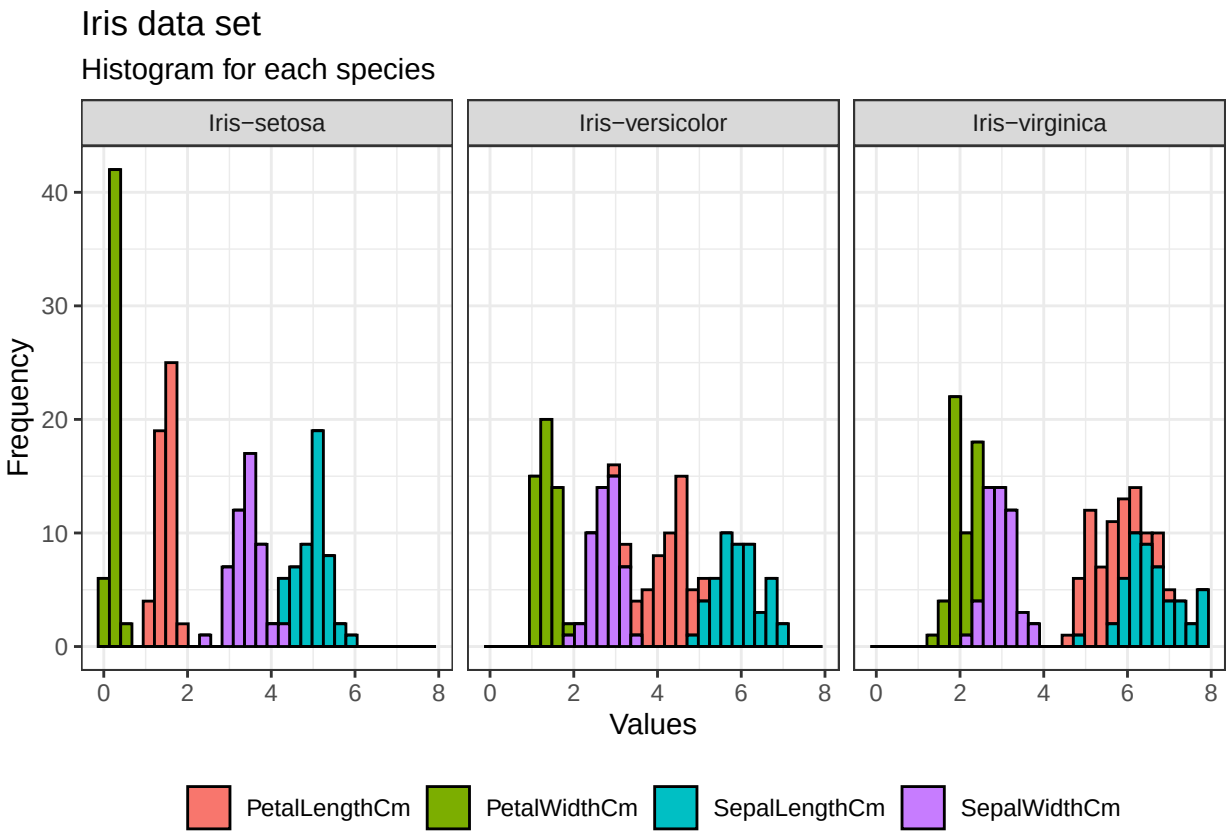



```
## Code for Histogram
iris_grouped <- group_by(iris, Species)

p6 <- iris_grouped %>% ggplot(aes(SepalLengthCm, fill = Species)) + geom_density(alpha = 0.2) +
theme_bw()

p7 <- iris_grouped %>% ggplot(aes(SepalWidthCm, fill = Species)) + geom_density(alpha = 0.2) +
theme_bw()

p8 <- iris_grouped %>% ggplot(aes(PetalLengthCm, fill = Species)) + geom_density(alpha = 0.2) +
theme_bw()

p9 <- iris_grouped %>% ggplot(aes(PetalWidthCm, fill = Species)) + geom_density(alpha = 0.2) +
theme_bw()

grid.arrange(p6,p7,p8,p9, ncol = 2)
```



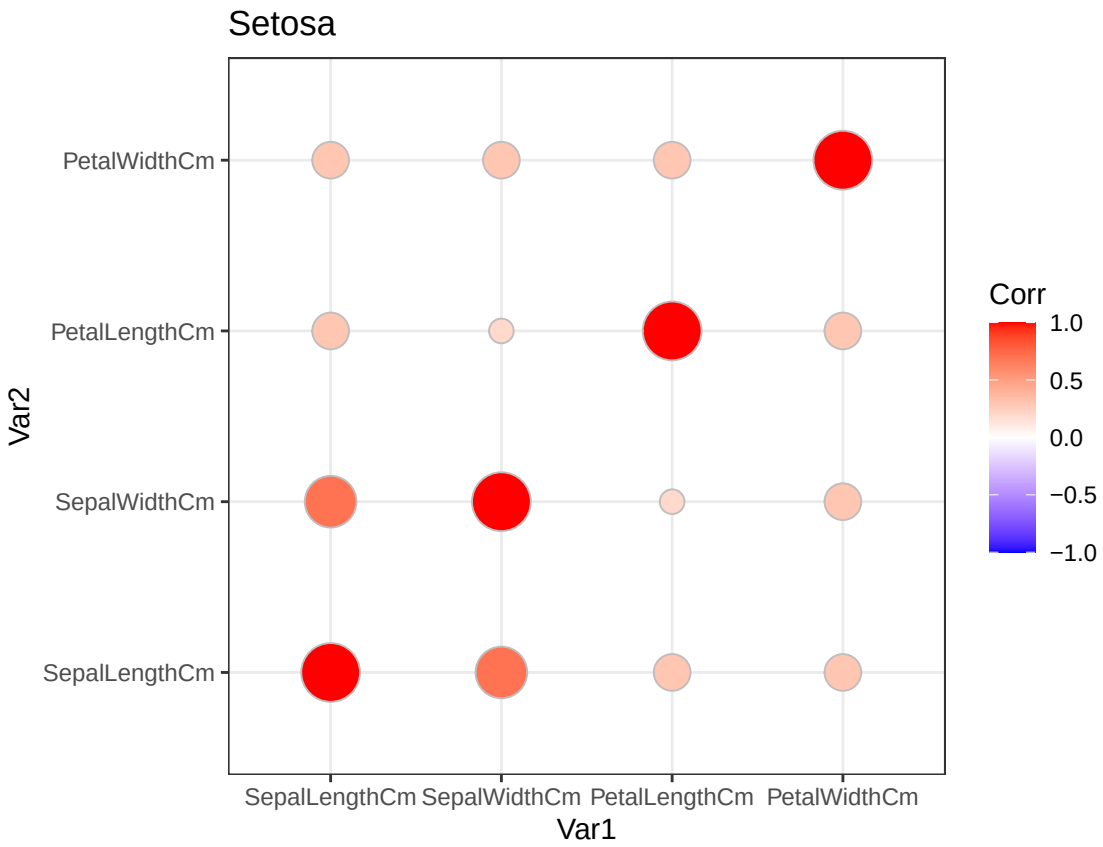

```
# Histogram for each species
p10 <- iris %>%
  gather(Attributes, Value, 1:4) %>%
  ggplot(aes(x= Value, fill= Attributes)) +
  geom_histogram(colour= "black") +
  facet_wrap(~Species) +
  theme_bw() +
  labs(x="Values", y="Frequency",
       title="Iris data set",
       subtitle="Histogram for each species") +
  theme(legend.title=element_blank(),
        legend.position="bottom")

p10
```

```
## `stat_bin()` using `bins = 30`. Pick better value with `binwidth`.
```

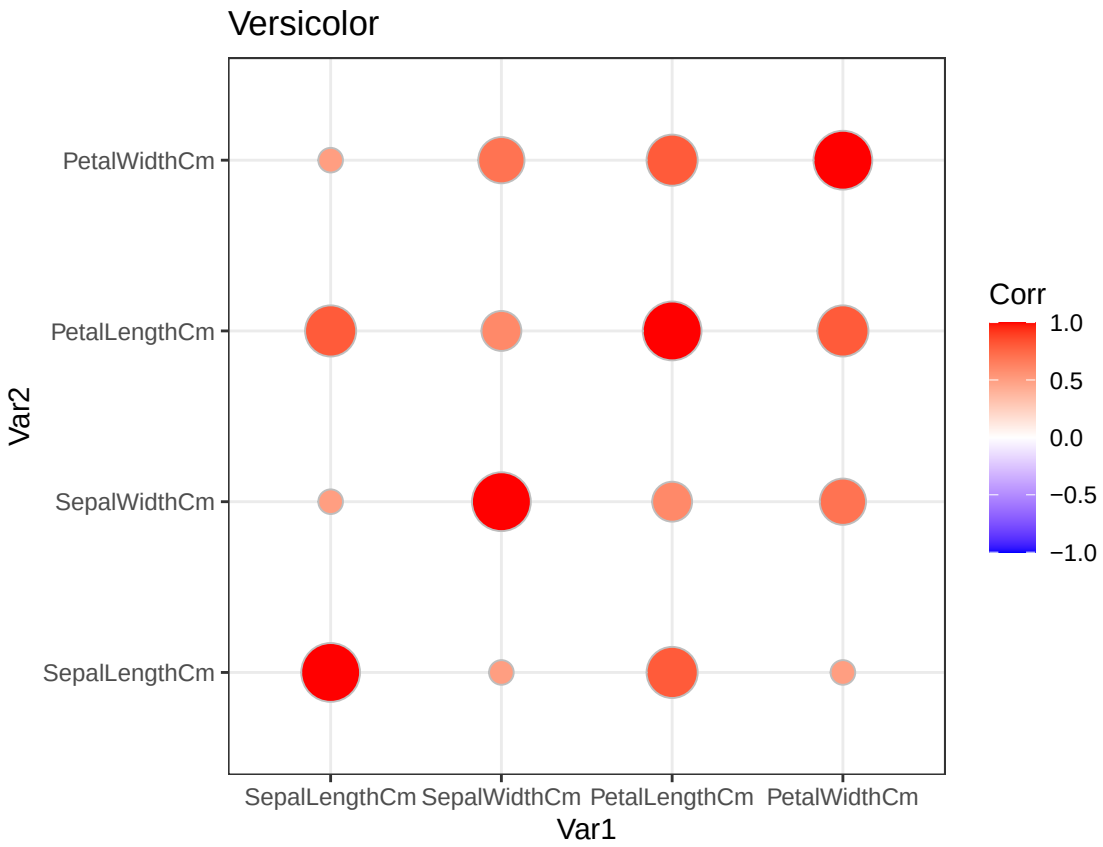


Now we will look into correlation plots

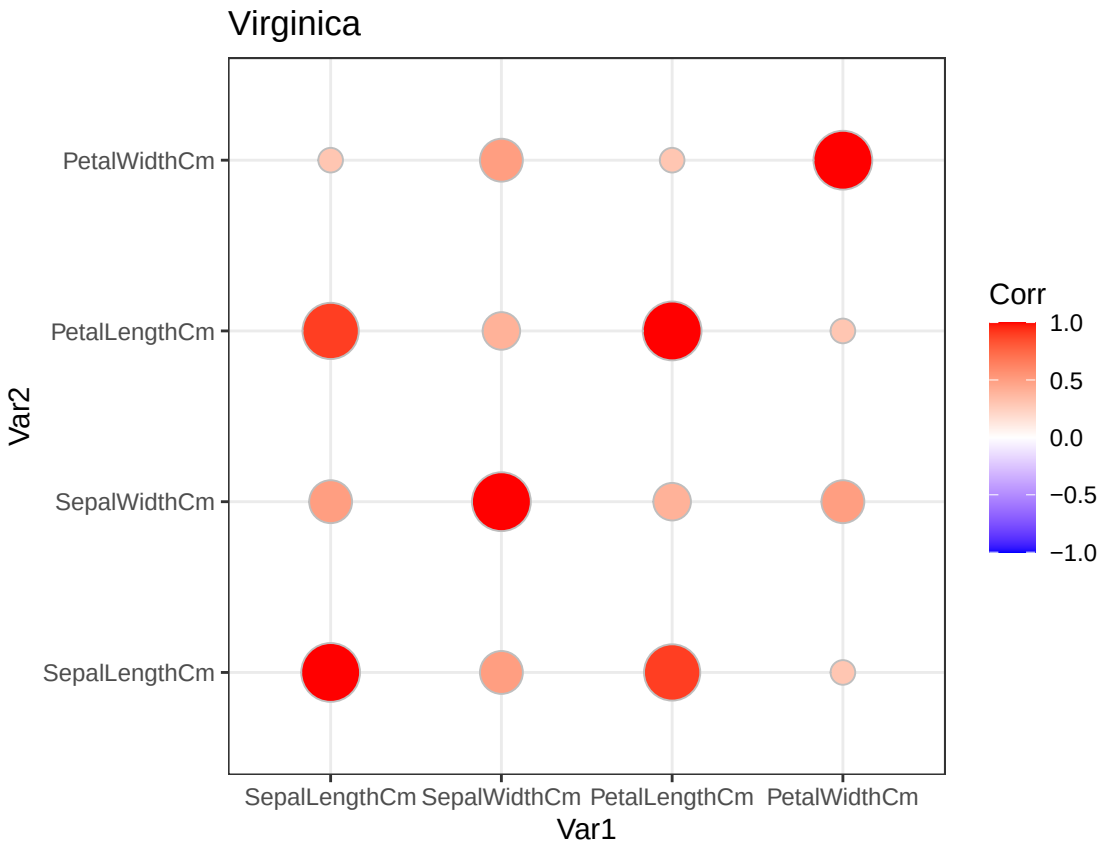
```
## Code for correlation plots
iris.setosa <- iris[iris$Species == 'Iris-setosa',]
iris.versicolor <- iris[iris$Species == 'Iris-versicolor',]
iris.virginica <- iris[iris$Species == 'Iris-virginica',]

corr1 <- round(cor(iris.setosa[,1:4]),1)
ggcorrplot(corr1, method = "circle") + theme_bw() + labs(title = "Setosa")
```




```
corr2 <- round(cor(iris.versicolor[,1:4]),1)
ggcorrplot(corr2, method = "circle") + theme_bw() + labs(title = "Versicolor")
```




```
corr3 <- round(cor(iris.virginica[,1:4]),1)
ggcorrplot(corr3, method = "circle") + theme_bw() + labs(title = "Virginica")
```



Now we will apply various machine algorithms for this classification problem. From the correlation plot, we can see that there is some correlation among the variables for all flower species. We also see that there is some separability among the variables based on the variables among the species. For example, Setosa has a much larger Sepal Lenth compared to the other two species. Ditto for Petal Length and Petal Width.

We get an idea from the plots that some of the classes are partially linearly separable in some dimensions, so we are expecting generally good results.

Let's evaluate 5 different algorithms:

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

The caret package allows you to quickly test ML algorithms before depoying them in a prodcution environment. Using the trainControl, we are telling R to perfrom 10-fold cross validation. Each model has ten results, and an accuracy distribution is obtained, which can then be compared to chose the best model. Caret does support tuning each model, but we're not doing that for this dataset

```
# Set seed for reproducibility
set.seed(123)

# Splitting the data with sample.split
sample = sample.split(iris$Species, SplitRatio = 0.80)
```

```r
# Subsetting the train and test data
train <- subset(iris, sample == TRUE)
test <- subset(iris, sample == FALSE)

dataset <- iris

# TrainControl setup
control <- trainControl(method="cv", number=10)
metric <- "Accuracy"

# a) linear algorithms
set.seed(90)
fit.lda <- train(Species~., data=dataset, method="lda", metric=metric, trControl=control)

# b) nonlinear algorithms
# CART
set.seed(90)
fit.cart <- train(Species~., data=dataset, method="rpart", metric=metric, trControl=control)

# kNN
set.seed(90)
fit.knn <- train(Species~., data=dataset, method="knn", metric=metric, trControl=control)

# c) advanced algorithms
# SVM
set.seed(90)
fit.svm <- train(Species~., data=dataset, method="svmRadial", metric=metric, trControl=control)

# Random Forest
set.seed(90)
fit.rf <- train(Species~., data=dataset, method="rf", metric=metric, trControl=control)

# Summarize accuracy of models
results <- resamples(list(lda=fit.lda, cart=fit.cart, knn=fit.knn, svm=fit.svm, rf=fit.rf))
summary(results)
```

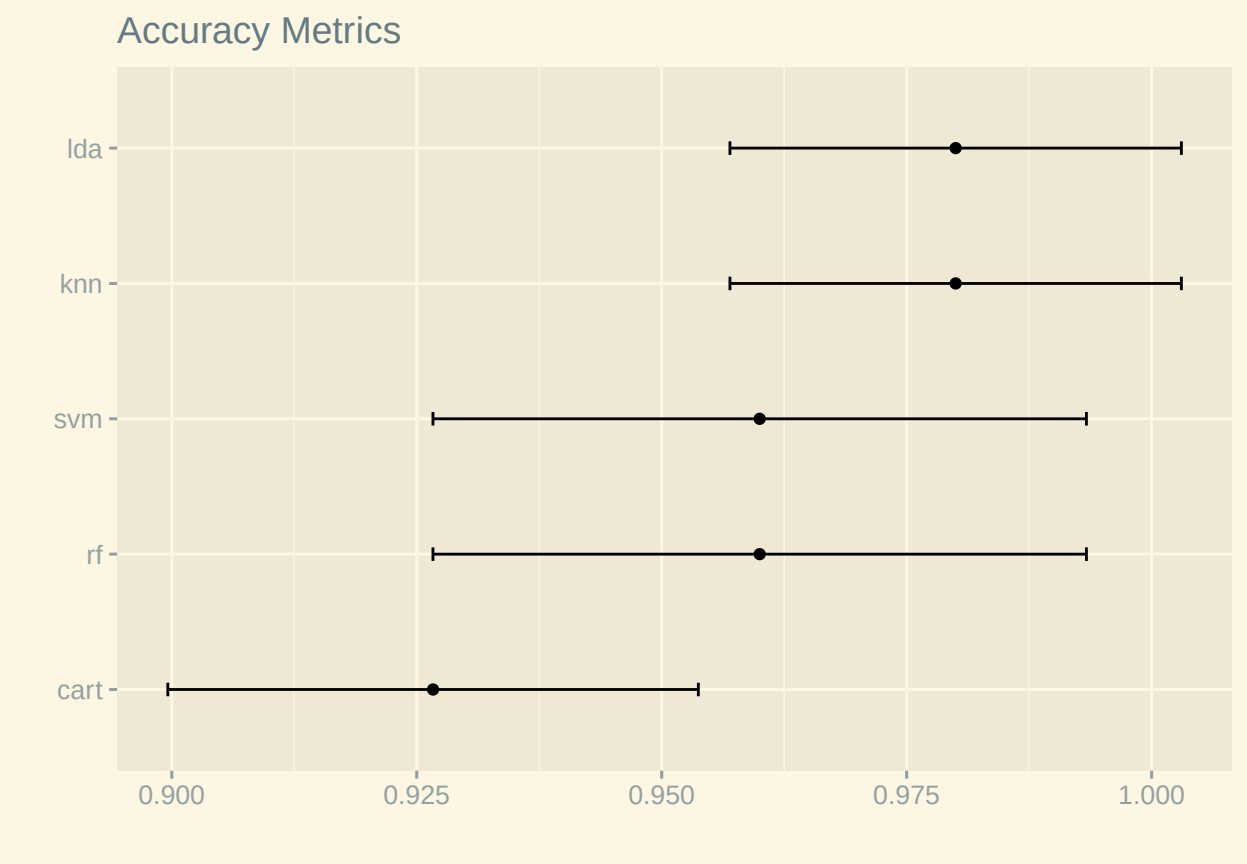
```
##
## Call:
## summary.resamples(object = results)
##
## Models: lda, cart, knn, svm, rf
## Number of resamples: 10
##
## Accuracy
##           Min.   1st Qu.    Median      Mean   3rd Qu. Max. NA's
## lda  0.9333333 0.9500000 1.0000000 0.9800000 1.0000000    1    0
## cart 0.8666667 0.9333333 0.9333333 0.9266667 0.9333333    1    0
## knn  0.9333333 0.9500000 1.0000000 0.9800000 1.0000000    1    0
## svm  0.8666667 0.9333333 0.9666667 0.9600000 1.0000000    1    0
## rf   0.8666667 0.9333333 0.9666667 0.9600000 1.0000000    1    0
##
## Kappa
##      Min. 1st Qu. Median Mean 3rd Qu. Max. NA's
```

```
## lda   0.9   0.925   1.00 0.97     1.0    1    0
## cart  0.8   0.900   0.90 0.89     0.9    1    0
## knn   0.9   0.925   1.00 0.97     1.0    1    0
## svm   0.8   0.900   0.95 0.94     1.0    1    0
## rf    0.8   0.900   0.95 0.94     1.0    1    0
```

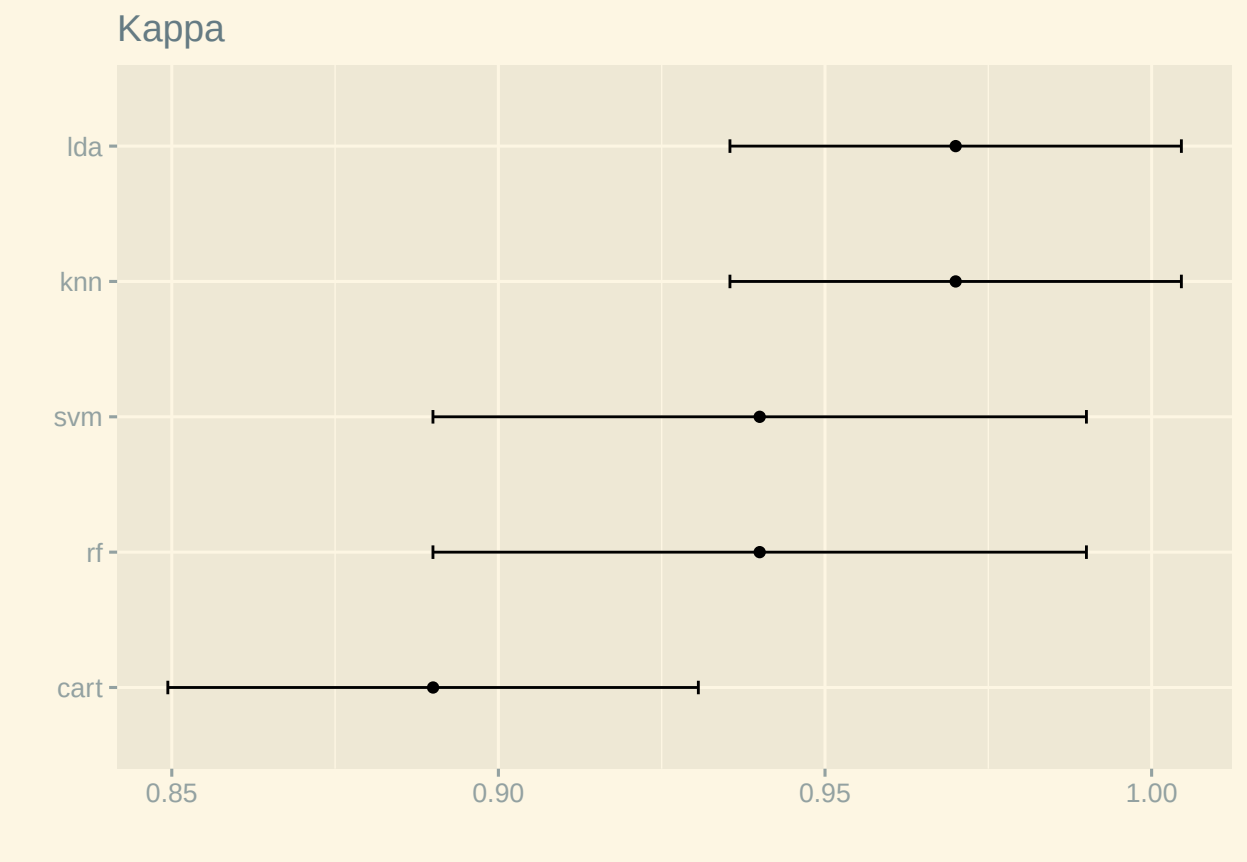
```
# Plot accuracy metrics
results <- resamples(list(lda=fit.lda, cart=fit.cart, knn=fit.knn, svm=fit.svm, rf=fit.rf))
summary(results)
```

```
##
## Call:
## summary.resamples(object = results)
##
## Models: lda, cart, knn, svm, rf
## Number of resamples: 10
##
## Accuracy
##           Min.   1st Qu.    Median      Mean   3rd Qu. Max. NA's
## lda  0.9333333 0.9500000 1.0000000 0.9800000 1.0000000    1    0
## cart 0.8666667 0.9333333 0.9333333 0.9266667 0.9333333    1    0
## knn  0.9333333 0.9500000 1.0000000 0.9800000 1.0000000    1    0
## svm  0.8666667 0.9333333 0.9666667 0.9600000 1.0000000    1    0
## rf   0.8666667 0.9333333 0.9666667 0.9600000 1.0000000    1    0
##
## Kappa
##      Min. 1st Qu. Median Mean 3rd Qu. Max. NA's
## lda   0.9   0.925   1.00 0.97     1.0    1    0
## cart  0.8   0.900   0.90 0.89     0.9    1    0
## knn   0.9   0.925   1.00 0.97     1.0    1    0
## svm   0.8   0.900   0.95 0.94     1.0    1    0
## rf    0.8   0.900   0.95 0.94     1.0    1    0
```

```
ggplot(data = results, mapping = NULL, metric = "Accuracy",
       output = "layered", environment = NULL)  + theme_solarized_2() +
  labs(title = "Accuracy Metrics", xlab = "Accuracy Values",
       ylab = "Method")
```




```
# Plot Kappa metrics
ggplot(data = results, mapping = NULL, metric = "Kappa",
       output = "layered", environment = NULL)  + theme_solarized_2() +
  labs(title = "Kappa", xlab = "Accuracy Values",
       ylab = "Method")
```



It turns out that LDA performs the best based on our training data. Now, let's apply LDA to the test dataset and obtain the predictions.

```
# estimate skill of LDA on the validation dataset
predictions <- predict(fit.lda, test)
# Compute confusion matrix
confusionMatrix(predictions, test$Species)
```

```
## Confusion Matrix and Statistics
##
##                  Reference
## Prediction        Iris-setosa Iris-versicolor Iris-virginica
##   Iris-setosa              10               0              0
##   Iris-versicolor           0               9              0
##   Iris-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: Iris-setosa Class: Iris-versicolor
## Sensitivity                      1.0000                 0.9000
## Specificity                      1.0000                 1.0000
## Pos Pred Value                   1.0000                 1.0000
## Neg Pred Value                   1.0000                 0.9524
## Prevalence                       0.3333                 0.3333
## Detection Rate                   0.3333                 0.3000
## Detection Prevalence             0.3333                 0.3000
## Balanced Accuracy                1.0000                 0.9500
##                      Class: Iris-virginica
## Sensitivity                         1.0000
## Specificity                         0.9500
## Pos Pred Value                      0.9091
## Neg Pred Value                      1.0000
## Prevalence                          0.3333
## Detection Rate                      0.3333
## Detection Prevalence                0.3667
## Balanced Accuracy                   0.9750
```

Conclusion:

In this project, we successfully applied various machine learning algorithms to classify iris species, with Linear Discriminant Analysis (LDA) emerging as the most accurate model. Through comprehensive data visualization and statistical analysis, we observed significant differences between species, contributing to the model's accuracy. With a final prediction accuracy of over 96%, the project demonstrates the effectiveness of classification techniques in solving this classic machine learning problem.