predict-iris-species

Sulaksh Swami

August 17, 2021

Synopsis

This predictive modeling project aims to build a machine learning model to predict the species of iris based on 4 numeric features. The data comes from the iris dataset built into R.

Setup

Import the libraries needed for the project.

```
library(e1071)
library(naniar)
library(ggplot2)
library(GGally)

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

library(tidyr)
library(corrplot)

corrplot 0.89 loaded

library(caret)

Loading required package: lattice
```

Load the Data

Load the data.

```
data(iris)
```

Store the data in a data frame.

```
df <- iris
```

Describe the Data

Look at the head and tail of the data.

```
head(df)
```

```
  Sepal.Length Sepal.Width Petal.Length Petal.Width Species
1          5.1         3.5          1.4         0.2  setosa
2          4.9         3.0          1.4         0.2  setosa
3          4.7         3.2          1.3         0.2  setosa
4          4.6         3.1          1.5         0.2  setosa
5          5.0         3.6          1.4         0.2  setosa
6          5.4         3.9          1.7         0.4  setosa
```

```
tail(df)
```

```
    Sepal.Length Sepal.Width Petal.Length Petal.Width   Species
145          6.7         3.3          5.7         2.5 virginica
146          6.7         3.0          5.2         2.3 virginica
147          6.3         2.5          5.0         1.9 virginica
148          6.5         3.0          5.2         2.0 virginica
149          6.2         3.4          5.4         2.3 virginica
150          5.9         3.0          5.1         1.8 virginica
```

Look at the dimensions of the data.

```
dim(df)
```

```
[1] 150   5
```

The dataset appears to have 150 rows and 5 columns.

Look at the data types of each variable in the data.

```
sapply(df, class)
```

```
Sepal.Length  Sepal.Width Petal.Length  Petal.Width      Species 
   "numeric"    "numeric"    "numeric"    "numeric"     "factor" 
```

Obtain descriptive statistics for the data.

```
summary(df)
```

```
  Sepal.Length    Sepal.Width     Petal.Length    Petal.Width   
 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   :3.057   Mean   :3.758   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  
 setosa    :50  
 versicolor:50  
 virginica :50  
```

Obtain the standard deviations of the numeric variables. All variables are numeric here.

```
X <- df[, colnames(df) != "Species"]

sapply(X, sd)

Sepal.Length  Sepal.Width Petal.Length  Petal.Width
   0.8280661    0.4358663    1.7652982    0.7622377
```

Obtain the distribution of instances across different class labels.

```
y <- df$Species

cbind(frequency = table(y),
      percentage = prop.table(table(y))*100)

           frequency percentage
setosa            50   33.33333
versicolor        50   33.33333
virginica         50   33.33333
```

Obtain the correlations between the numeric variables.

```
cor(X)

             Sepal.Length Sepal.Width Petal.Length Petal.Width
Sepal.Length    1.0000000  -0.1175698    0.8717538   0.8179411
Sepal.Width    -0.1175698   1.0000000   -0.4284401  -0.3661259
Petal.Length    0.8717538  -0.4284401    1.0000000   0.9628654
Petal.Width     0.8179411  -0.3661259    0.9628654   1.0000000
```

Obtain the skew of each numeric variable in the data.

```
sapply(X, skewness)

Sepal.Length  Sepal.Width Petal.Length  Petal.Width
   0.3086407    0.3126147   -0.2694109   -0.1009166
```

Use the Shapiro-Wilk test to check if the numeric variables in the data are Gaussian.

```
# Use a significance level of 0.05

p.values <- as.numeric(sapply(X, shapiro.test)["p.value", ])
is.gaussian <- (p.values >= 0.05)

check.gaussian <- data.frame(p.values = p.values,
                             is.gaussian = is.gaussian)
rownames(check.gaussian) <- colnames(X)

check.gaussian
```

```
                 p.values is.gaussian
Sepal.Length 1.018116e-02       FALSE
Sepal.Width  1.011543e-01        TRUE
Petal.Length 7.412263e-10       FALSE
Petal.Width  1.680465e-08       FALSE
```

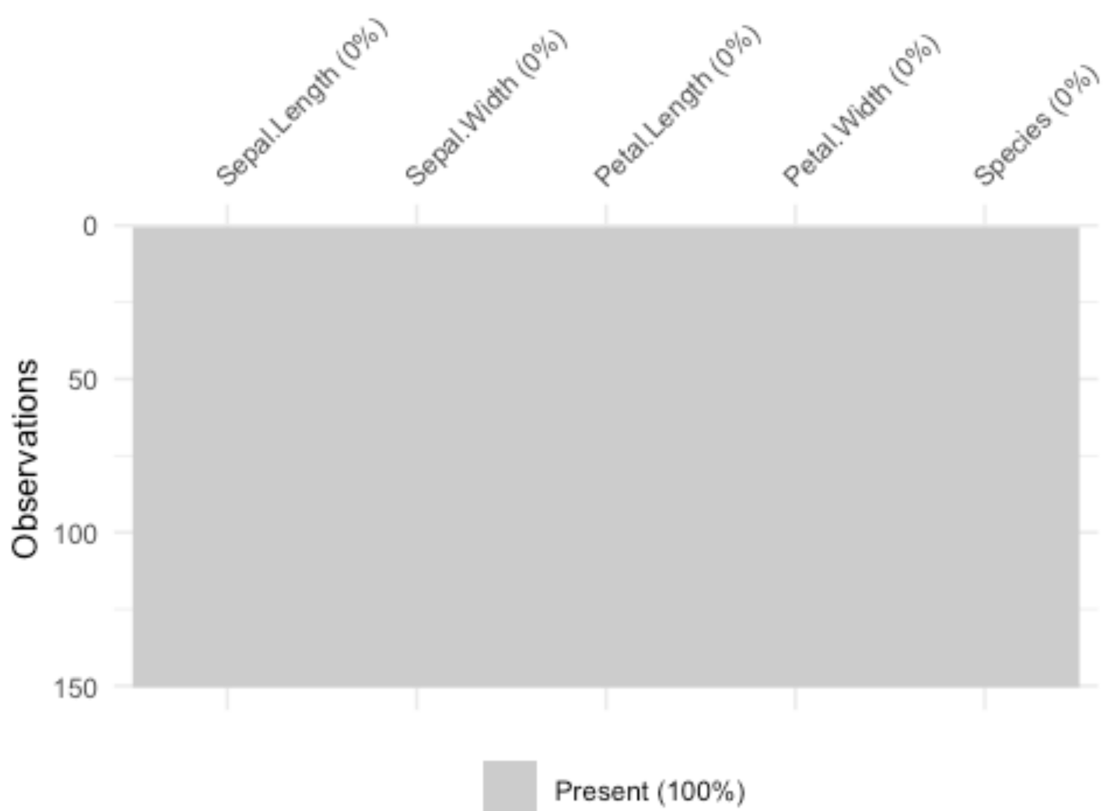
The output indicates that only Sepal.Width is Gaussian at a 0.05 significance level. Keeping this in mind, I pick the XGBoost algorithm for modeling the problem, since this algorithm doesn't assume that its features are Gaussian.

Visualize the Data

Univariate Plots

Make a missing value plot to diagnose the presence of missing values in the data.

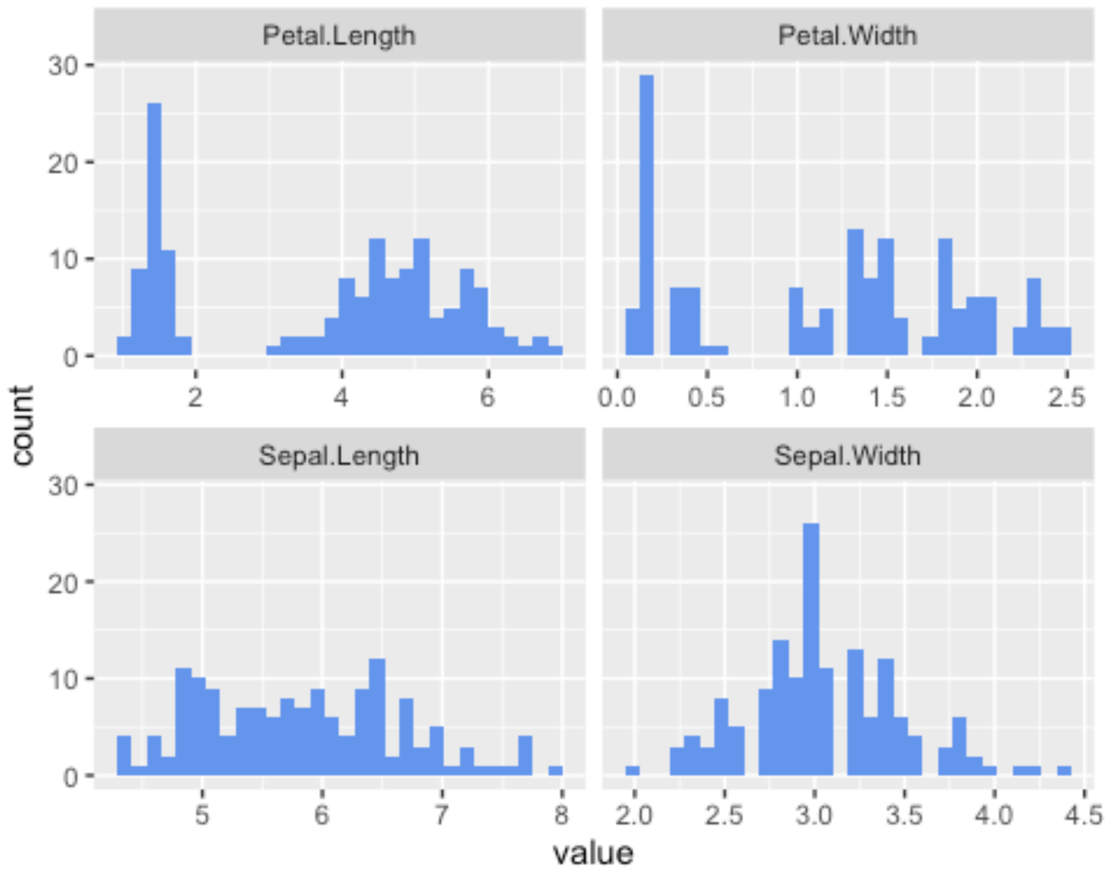
```
vis_miss(df)
```



The plot's output indicates that no variable has missing values. Hence, no imputation will have to be carried out during preprocessing.

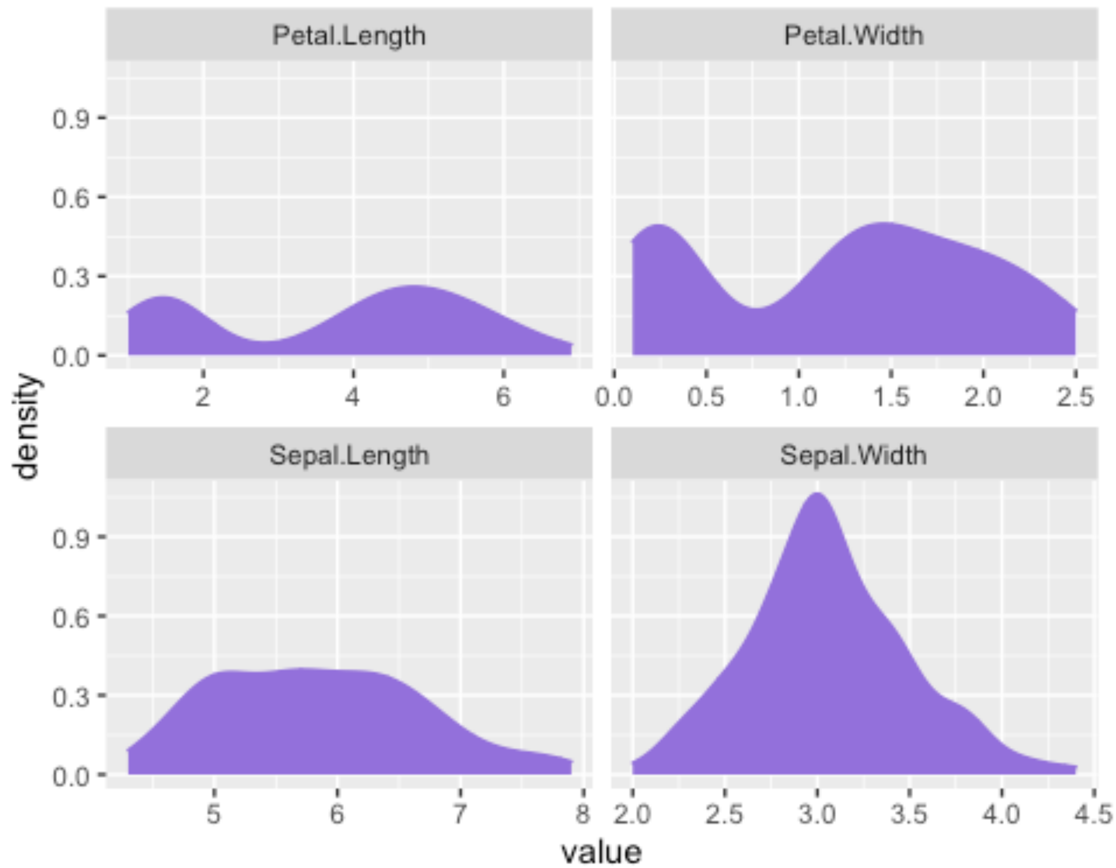
Make a histogram for each numeric variable.

```
ggplot(data = gather(X)) +
       geom_histogram(mapping = aes(x = value),
                      fill = "cornflowerblue") +
       facet_wrap(~key,
                  scales = "free_x")

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



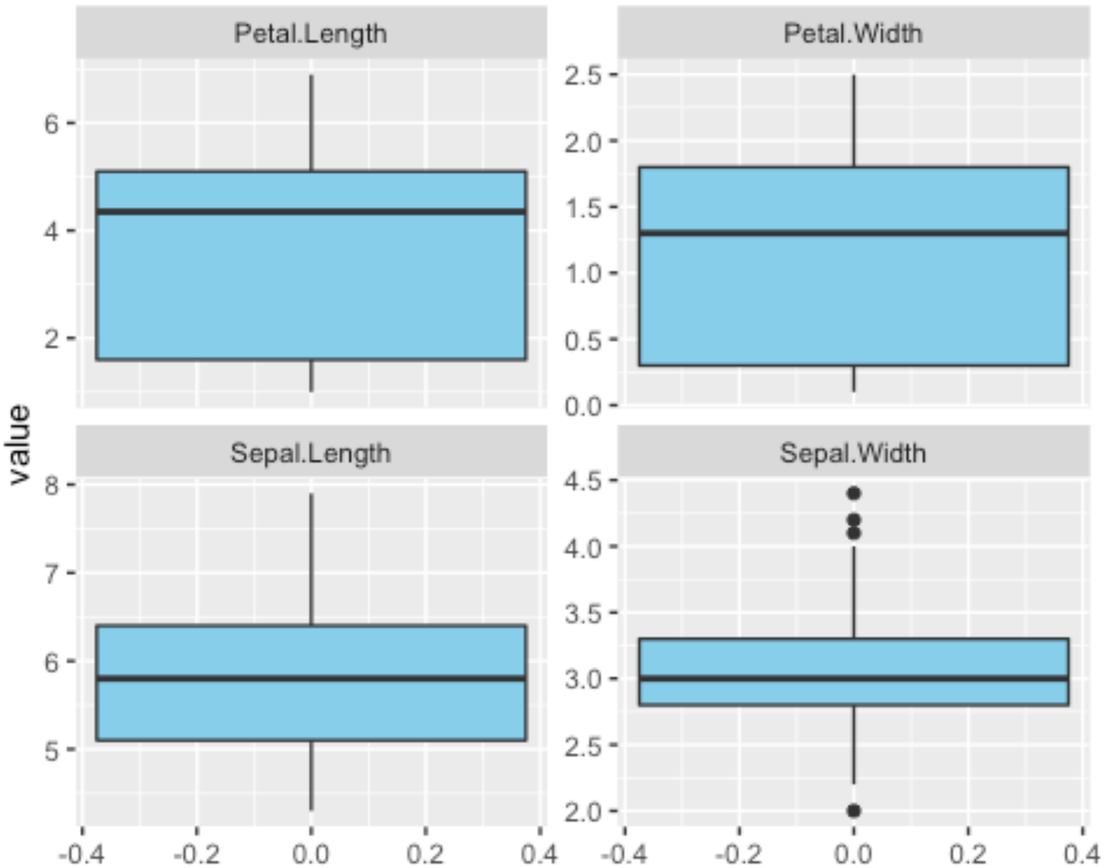
Make a density plot for each variable.

```
ggplot(data = gather(X)) +
       geom_density(mapping = aes(x = value),
                    color = "mediumpurple",
                    fill = "mediumpurple") +
       facet_wrap(~key,
                  scales = "free_x")
```



Make a boxplot for each variable.

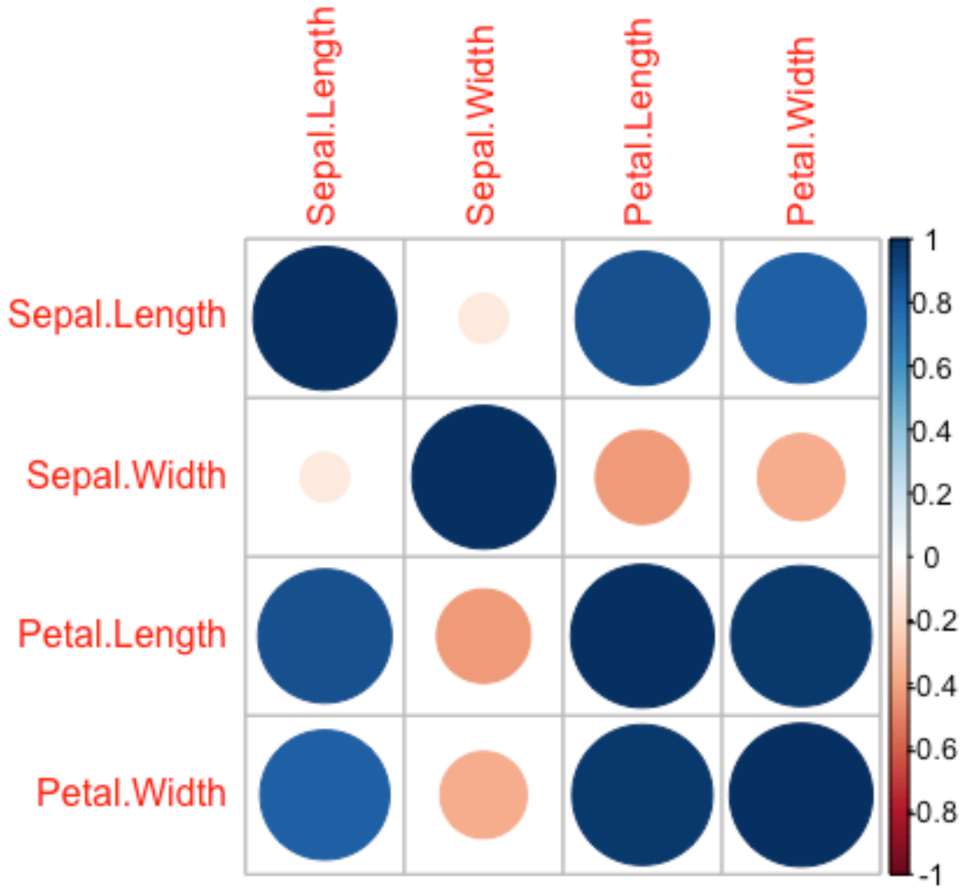
```
ggplot(data = gather(X)) +
        geom_boxplot(mapping = aes(y = value),
                     fill = "skyblue") +
        facet_wrap(~key,
                   scales = "free_y")
```



Multivariate Plots

Make a correlation matrix plot to visualize the correlation between the numeric variables in the data.

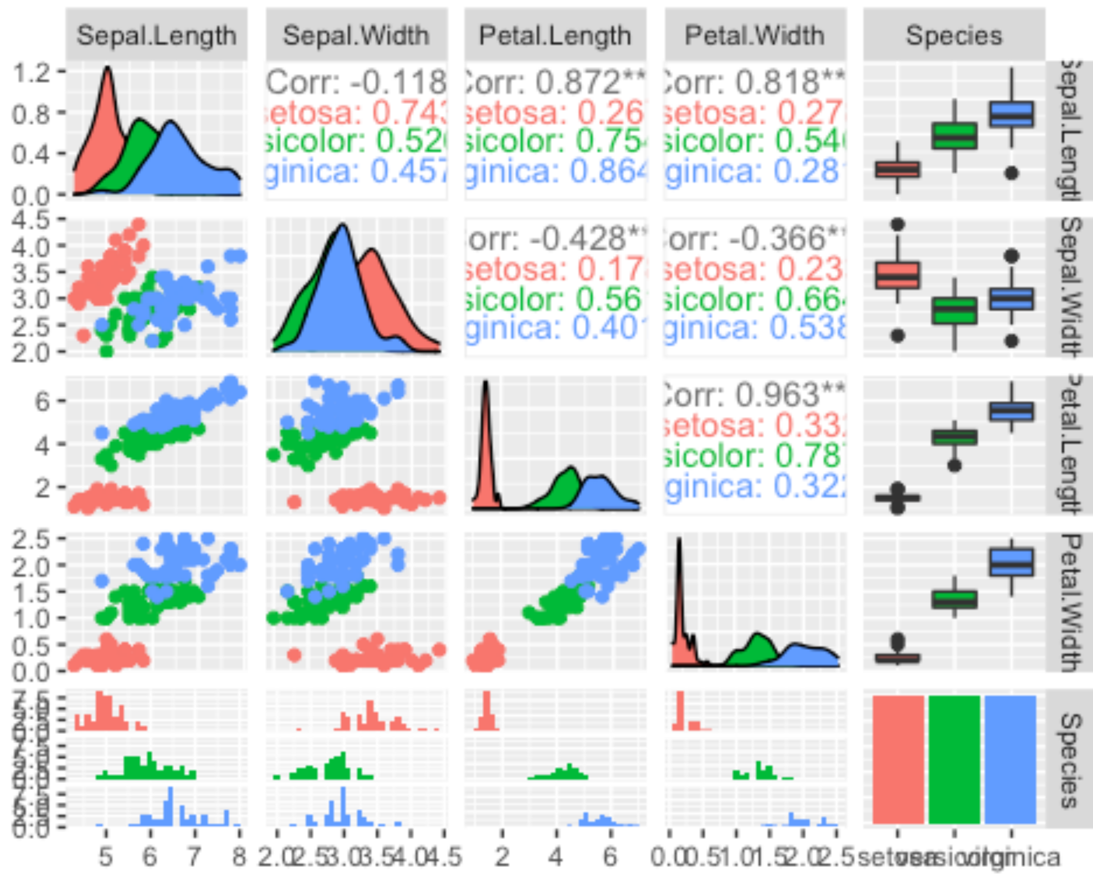
```
correlations <- cor(X)

corrplot(correlations,
         method = "circle")
```



Make a scatter plot matrix for the data frame.

```
ggpairs(data = df,
        aes(color = Species))
```

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



Data Partitioning

Data partitioning of the data frame into the features and the target variable has already been done.

```
str(X)

'data.frame':   150 obs. of  4 variables:
 $ Sepal.Length: num  5.1 4.9 4.7 4.6 5 5.4 4.6 5 4.4 4.9 ...
 $ Sepal.Width : num  3.5 3 3.2 3.1 3.6 3.9 3.4 3.4 2.9 3.1 ...
 $ Petal.Length: num  1.4 1.4 1.3 1.5 1.4 1.7 1.4 1.5 1.4 1.5 ...
 $ Petal.Width : num  0.2 0.2 0.2 0.2 0.2 0.4 0.3 0.2 0.2 0.1 ...

str(y)

 Factor w/ 3 levels "setosa","versicolor",..: 1 1 1 1 1 1 1 1 1 1 ...
```

Preprocessing

All variables are numeric. The data needs to be scaled to a mean of 0 and a standard deviation of 1. Preprocessing will be carried out during training.

Tune the Model Parameters

Make a grid of candidate parameters.

```
tune.grid <- expand.grid(nrounds = 100,
                         eta = c(0.02, 0.04, 0.06, 0.08, 0.1),
                         subsample = c(0.5, 0.75, 1),
                         colsample_bytree = c(0.4, 0.6, 0.8, 1),
                         max_depth = c(4, 6, 8, 10),
                         min_child_weight = 0,
                         gamma = 0)
```

Set up cross-validation.

```
train.control <- trainControl(method = "cv",
                              number = 5)
```

Train the model using the grid search harness.

```
model <- train(x = X,
               y = y,
               method = "xgbTree",
               preProcess = c("center", "scale"),
               metric = "Accuracy",
               maximize = TRUE,
               trControl = train.control,
               tuneGrid = tune.grid)

model

eXtreme Gradient Boosting

150 samples
  4 predictor
  3 classes: 'setosa', 'versicolor', 'virginica'

Pre-processing: centered (4), scaled (4)
Resampling: Cross-Validated (5 fold)
Summary of sample sizes: 120, 120, 120, 120, 120
Resampling results across tuning parameters:

  eta   max_depth  colsample_bytree  subsample  Accuracy   Kappa
  0.02   4         0.4               0.50       0.9466667  0.92
  0.02   4         0.4               0.75       0.9466667  0.92
  0.02   4         0.4               1.00       0.9333333  0.90
  0.02   4         0.6               0.50       0.9533333  0.93
  0.02   4         0.6               0.75       0.9533333  0.93
  0.02   4         0.6               1.00       0.9533333  0.93
  0.02   4         0.8               0.50       0.9533333  0.93
  0.02   4         0.8               0.75       0.9600000  0.94
  0.02   4         0.8               1.00       0.9600000  0.94
```

0.02	4	1.0	0.50	0.9533333	0.93
0.02	4	1.0	0.75	0.9533333	0.93
0.02	4	1.0	1.00	0.9533333	0.93
0.02	6	0.4	0.50	0.9333333	0.90
0.02	6	0.4	0.75	0.9400000	0.91
0.02	6	0.4	1.00	0.9266667	0.89
0.02	6	0.6	0.50	0.9533333	0.93
0.02	6	0.6	0.75	0.9533333	0.93
0.02	6	0.6	1.00	0.9533333	0.93
0.02	6	0.8	0.50	0.9533333	0.93
0.02	6	0.8	0.75	0.9600000	0.94
0.02	6	0.8	1.00	0.9600000	0.94
0.02	6	1.0	0.50	0.9533333	0.93
0.02	6	1.0	0.75	0.9533333	0.93
0.02	6	1.0	1.00	0.9533333	0.93
0.02	8	0.4	0.50	0.9400000	0.91
0.02	8	0.4	0.75	0.9400000	0.91
0.02	8	0.4	1.00	0.9466667	0.92
0.02	8	0.6	0.50	0.9533333	0.93
0.02	8	0.6	0.75	0.9533333	0.93
0.02	8	0.6	1.00	0.9600000	0.94
0.02	8	0.8	0.50	0.9533333	0.93
0.02	8	0.8	0.75	0.9600000	0.94
0.02	8	0.8	1.00	0.9600000	0.94
0.02	8	1.0	0.50	0.9533333	0.93
0.02	8	1.0	0.75	0.9533333	0.93
0.02	8	1.0	1.00	0.9533333	0.93
0.02	10	0.4	0.50	0.9400000	0.91
0.02	10	0.4	0.75	0.9466667	0.92
0.02	10	0.4	1.00	0.9266667	0.89
0.02	10	0.6	0.50	0.9466667	0.92
0.02	10	0.6	0.75	0.9533333	0.93
0.02	10	0.6	1.00	0.9533333	0.93
0.02	10	0.8	0.50	0.9533333	0.93
0.02	10	0.8	0.75	0.9600000	0.94
0.02	10	0.8	1.00	0.9600000	0.94
0.02	10	1.0	0.50	0.9466667	0.92
0.02	10	1.0	0.75	0.9533333	0.93
0.02	10	1.0	1.00	0.9533333	0.93
0.04	4	0.4	0.50	0.9533333	0.93
0.04	4	0.4	0.75	0.9400000	0.91
0.04	4	0.4	1.00	0.9333333	0.90
0.04	4	0.6	0.50	0.9533333	0.93
0.04	4	0.6	0.75	0.9600000	0.94
0.04	4	0.6	1.00	0.9600000	0.94
0.04	4	0.8	0.50	0.9533333	0.93
0.04	4	0.8	0.75	0.9600000	0.94
0.04	4	0.8	1.00	0.9533333	0.93
0.04	4	1.0	0.50	0.9533333	0.93
0.04	4	1.0	0.75	0.9533333	0.93

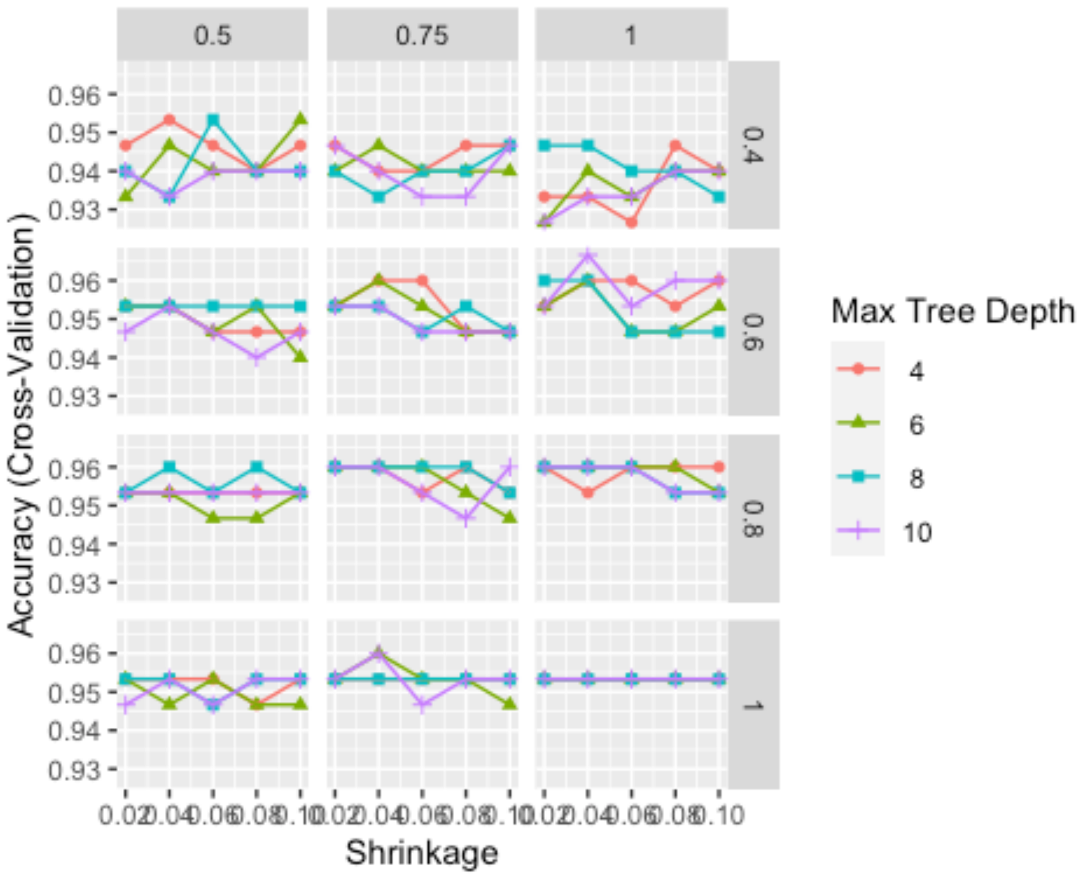
0.04	4	1.0	1.00	0.9533333	0.93
0.04	6	0.4	0.50	0.9466667	0.92
0.04	6	0.4	0.75	0.9466667	0.92
0.04	6	0.4	1.00	0.9400000	0.91
0.04	6	0.6	0.50	0.9533333	0.93
0.04	6	0.6	0.75	0.9600000	0.94
0.04	6	0.6	1.00	0.9600000	0.94
0.04	6	0.8	0.50	0.9533333	0.93
0.04	6	0.8	0.75	0.9600000	0.94
0.04	6	0.8	1.00	0.9600000	0.94
0.04	6	1.0	0.50	0.9466667	0.92
0.04	6	1.0	0.75	0.9600000	0.94
0.04	6	1.0	1.00	0.9533333	0.93
0.04	8	0.4	0.50	0.9333333	0.90
0.04	8	0.4	0.75	0.9333333	0.90
0.04	8	0.4	1.00	0.9466667	0.92
0.04	8	0.6	0.50	0.9533333	0.93
0.04	8	0.6	0.75	0.9533333	0.93
0.04	8	0.6	1.00	0.9600000	0.94
0.04	8	0.8	0.50	0.9600000	0.94
0.04	8	0.8	0.75	0.9600000	0.94
0.04	8	0.8	1.00	0.9600000	0.94
0.04	8	1.0	0.50	0.9533333	0.93
0.04	8	1.0	0.75	0.9533333	0.93
0.04	8	1.0	1.00	0.9533333	0.93
0.04	10	0.4	0.50	0.9333333	0.90
0.04	10	0.4	0.75	0.9400000	0.91
0.04	10	0.4	1.00	0.9333333	0.90
0.04	10	0.6	0.50	0.9533333	0.93
0.04	10	0.6	0.75	0.9533333	0.93
0.04	10	0.6	1.00	0.9666667	0.95
0.04	10	0.8	0.50	0.9533333	0.93
0.04	10	0.8	0.75	0.9600000	0.94
0.04	10	0.8	1.00	0.9600000	0.94
0.04	10	1.0	0.50	0.9533333	0.93
0.04	10	1.0	0.75	0.9600000	0.94
0.04	10	1.0	1.00	0.9533333	0.93
0.06	4	0.4	0.50	0.9466667	0.92
0.06	4	0.4	0.75	0.9400000	0.91
0.06	4	0.4	1.00	0.9266667	0.89
0.06	4	0.6	0.50	0.9466667	0.92
0.06	4	0.6	0.75	0.9600000	0.94
0.06	4	0.6	1.00	0.9600000	0.94
0.06	4	0.8	0.50	0.9533333	0.93
0.06	4	0.8	0.75	0.9533333	0.93
0.06	4	0.8	1.00	0.9600000	0.94
0.06	4	1.0	0.50	0.9533333	0.93
0.06	4	1.0	0.75	0.9533333	0.93
0.06	4	1.0	1.00	0.9533333	0.93
0.06	6	0.4	0.50	0.9400000	0.91

0.06	6	0.4	0.75	0.9400000	0.91
0.06	6	0.4	1.00	0.9333333	0.90
0.06	6	0.6	0.50	0.9466667	0.92
0.06	6	0.6	0.75	0.9533333	0.93
0.06	6	0.6	1.00	0.9466667	0.92
0.06	6	0.8	0.50	0.9466667	0.92
0.06	6	0.8	0.75	0.9600000	0.94
0.06	6	0.8	1.00	0.9600000	0.94
0.06	6	1.0	0.50	0.9533333	0.93
0.06	6	1.0	0.75	0.9533333	0.93
0.06	6	1.0	1.00	0.9533333	0.93
0.06	8	0.4	0.50	0.9533333	0.93
0.06	8	0.4	0.75	0.9400000	0.91
0.06	8	0.4	1.00	0.9400000	0.91
0.06	8	0.6	0.50	0.9533333	0.93
0.06	8	0.6	0.75	0.9466667	0.92
0.06	8	0.6	1.00	0.9466667	0.92
0.06	8	0.8	0.50	0.9533333	0.93
0.06	8	0.8	0.75	0.9600000	0.94
0.06	8	0.8	1.00	0.9600000	0.94
0.06	8	1.0	0.50	0.9466667	0.92
0.06	8	1.0	0.75	0.9533333	0.93
0.06	8	1.0	1.00	0.9533333	0.93
0.06	10	0.4	0.50	0.9400000	0.91
0.06	10	0.4	0.75	0.9333333	0.90
0.06	10	0.4	1.00	0.9333333	0.90
0.06	10	0.6	0.50	0.9466667	0.92
0.06	10	0.6	0.75	0.9466667	0.92
0.06	10	0.6	1.00	0.9533333	0.93
0.06	10	0.8	0.50	0.9533333	0.93
0.06	10	0.8	0.75	0.9533333	0.93
0.06	10	0.8	1.00	0.9600000	0.94
0.06	10	1.0	0.50	0.9466667	0.92
0.06	10	1.0	0.75	0.9466667	0.92
0.06	10	1.0	1.00	0.9533333	0.93
0.08	4	0.4	0.50	0.9400000	0.91
0.08	4	0.4	0.75	0.9466667	0.92
0.08	4	0.4	1.00	0.9466667	0.92
0.08	4	0.6	0.50	0.9466667	0.92
0.08	4	0.6	0.75	0.9466667	0.92
0.08	4	0.6	1.00	0.9533333	0.93
0.08	4	0.8	0.50	0.9533333	0.93
0.08	4	0.8	0.75	0.9600000	0.94
0.08	4	0.8	1.00	0.9600000	0.94
0.08	4	1.0	0.50	0.9466667	0.92
0.08	4	1.0	0.75	0.9533333	0.93
0.08	4	1.0	1.00	0.9533333	0.93
0.08	6	0.4	0.50	0.9400000	0.91
0.08	6	0.4	0.75	0.9400000	0.91
0.08	6	0.4	1.00	0.9400000	0.91

0.08	6	0.6	0.50	0.9533333	0.93
0.08	6	0.6	0.75	0.9466667	0.92
0.08	6	0.6	1.00	0.9466667	0.92
0.08	6	0.8	0.50	0.9466667	0.92
0.08	6	0.8	0.75	0.9533333	0.93
0.08	6	0.8	1.00	0.9600000	0.94
0.08	6	1.0	0.50	0.9466667	0.92
0.08	6	1.0	0.75	0.9533333	0.93
0.08	6	1.0	1.00	0.9533333	0.93
0.08	8	0.4	0.50	0.9400000	0.91
0.08	8	0.4	0.75	0.9400000	0.91
0.08	8	0.4	1.00	0.9400000	0.91
0.08	8	0.6	0.50	0.9533333	0.93
0.08	8	0.6	0.75	0.9533333	0.93
0.08	8	0.6	1.00	0.9466667	0.92
0.08	8	0.8	0.50	0.9600000	0.94
0.08	8	0.8	0.75	0.9600000	0.94
0.08	8	0.8	1.00	0.9533333	0.93
0.08	8	1.0	0.50	0.9533333	0.93
0.08	8	1.0	0.75	0.9533333	0.93
0.08	8	1.0	1.00	0.9533333	0.93
0.08	10	0.4	0.50	0.9400000	0.91
0.08	10	0.4	0.75	0.9333333	0.90
0.08	10	0.4	1.00	0.9400000	0.91
0.08	10	0.6	0.50	0.9400000	0.91
0.08	10	0.6	0.75	0.9466667	0.92
0.08	10	0.6	1.00	0.9600000	0.94
0.08	10	0.8	0.50	0.9533333	0.93
0.08	10	0.8	0.75	0.9466667	0.92
0.08	10	0.8	1.00	0.9533333	0.93
0.08	10	1.0	0.50	0.9533333	0.93
0.08	10	1.0	0.75	0.9533333	0.93
0.08	10	1.0	1.00	0.9533333	0.93
0.10	4	0.4	0.50	0.9466667	0.92
0.10	4	0.4	0.75	0.9466667	0.92
0.10	4	0.4	1.00	0.9400000	0.91
0.10	4	0.6	0.50	0.9466667	0.92
0.10	4	0.6	0.75	0.9466667	0.92
0.10	4	0.6	1.00	0.9600000	0.94
0.10	4	0.8	0.50	0.9533333	0.93
0.10	4	0.8	0.75	0.9533333	0.93
0.10	4	0.8	1.00	0.9600000	0.94
0.10	4	1.0	0.50	0.9533333	0.93
0.10	4	1.0	0.75	0.9533333	0.93
0.10	4	1.0	1.00	0.9533333	0.93
0.10	6	0.4	0.50	0.9533333	0.93
0.10	6	0.4	0.75	0.9400000	0.91
0.10	6	0.4	1.00	0.9400000	0.91
0.10	6	0.6	0.50	0.9400000	0.91
0.10	6	0.6	0.75	0.9466667	0.92

```
  0.10   6         0.6               1.00      0.9533333  0.93
  0.10   6         0.8               0.50      0.9533333  0.93
  0.10   6         0.8               0.75      0.9466667  0.92
  0.10   6         0.8               1.00      0.9533333  0.93
  0.10   6         1.0               0.50      0.9466667  0.92
  0.10   6         1.0               0.75      0.9466667  0.92
  0.10   6         1.0               1.00      0.9533333  0.93
  0.10   8         0.4               0.50      0.9400000  0.91
  0.10   8         0.4               0.75      0.9466667  0.92
  0.10   8         0.4               1.00      0.9333333  0.90
  0.10   8         0.6               0.50      0.9533333  0.93
  0.10   8         0.6               0.75      0.9466667  0.92
  0.10   8         0.6               1.00      0.9466667  0.92
  0.10   8         0.8               0.50      0.9533333  0.93
  0.10   8         0.8               0.75      0.9533333  0.93
  0.10   8         0.8               1.00      0.9533333  0.93
  0.10   8         1.0               0.50      0.9533333  0.93
  0.10   8         1.0               0.75      0.9533333  0.93
  0.10   8         1.0               1.00      0.9533333  0.93
  0.10  10         0.4               0.50      0.9400000  0.91
  0.10  10         0.4               0.75      0.9466667  0.92
  0.10  10         0.4               1.00      0.9400000  0.91
  0.10  10         0.6               0.50      0.9466667  0.92
  0.10  10         0.6               0.75      0.9466667  0.92
  0.10  10         0.6               1.00      0.9600000  0.94
  0.10  10         0.8               0.50      0.9533333  0.93
  0.10  10         0.8               0.75      0.9600000  0.94
  0.10  10         0.8               1.00      0.9533333  0.93
  0.10  10         1.0               0.50      0.9533333  0.93
  0.10  10         1.0               0.75      0.9533333  0.93
  0.10  10         1.0               1.00      0.9533333  0.93

Tuning parameter 'nrounds' was held constant at a value of 100
Tuning
 parameter 'gamma' was held constant at a value of 0
Tuning
 parameter 'min_child_weight' was held constant at a value of 0
Accuracy was used to select the optimal model using the largest value.
The final values used for the model were nrounds = 100, max_depth = 10, eta
 = 0.04, gamma = 0, colsample_bytree = 0.6, min_child_weight = 0 and
 subsample = 1.
```

```
ggplot(data = model)
```




```
model$bestTune

   nrounds max_depth  eta gamma colsample_bytree min_child_weight subsample
90     100        10 0.04     0              0.6                0         1

print("The best parameters obtained from the grid search are:")

[1] "The best parameters obtained from the grid search are:"

print(model$bestTune)

   nrounds max_depth  eta gamma colsample_bytree min_child_weight subsample
90     100        10 0.04     0              0.6                0         1

best.row <- rownames(model$bestTune)
best.accuracy <- model$results[best.row, ]$Accuracy
best.accuracy.standard.error <- model$results[best.row, ]$AccuracySD

cat("The accuracy score obtained for the best parameters is", best.accuracy,
"with a standard error of", best.accuracy.standard.error)

The accuracy score obtained for the best parameters is 0.9666667 with a
standard error of 0.02357023
```

Store the final model as a fit object.

```r
final.model <- model$finalModel
```

Save the Model

Save the model to disk so that it may be conveniently loaded later, as and when it is required to make predictions on iris data.

```r
saveRDS(object = final.model,
        file = "iris-model.rds")

dir()

[1] "iris-model.rds"              "predict-iris-species_files"
[3] "predict-iris-species.docx"   "predict-iris-species.Rmd"
```