Iris Predictive Analysis: A Species-Specific Model Exploration

Pritam Naskar

Introduction:

The Iris dataset is one of the most famous datasets in the field of **machine learning** and **statistics**. It was introduced by the British biologist and statistician **Ronald A. Fisher** in his 1936 paper **“The Use of Multiple Measurements in Taxonomic Problems”** as an example of discriminant analysis.

Import Libraries:

```
library(datasets)
library(dplyr)
library(ggplot2)
library(corrplot)
library(kableExtra)
library(gridExtra)
library(GGally)
library(nnet)
library(randomForest)
library(caret)
library(MASSExtra)
library(car)
library(shiny)
```

Import Dataset:

The Iris dataset is included in the `datasets` package in R, making it readily available for analysis. To load the Iris dataset in R:

```
data(iris)
head(iris)
```

```
##   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
```

Descriptive Statistic:

Here is the overview of the Iris dataset:

```
cat("This dataset has", nrow(iris), "rows and", ncol(iris), "columns.\n")
```

```
## This dataset has 150 rows and 5 columns.
```

```
str(iris)
```

```
## 'data.frame':    150 obs. of  5 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 ...
##  $ Species     : Factor w/ 3 levels "setosa","versicolor",..: 1 1 1 1 1 1 1 1 1 1 ...
```

```
options(width = 100)
summary(iris)
```

```
##   Sepal.Length    Sepal.Width     Petal.Length    Petal.Width          Species
##  Min.   :4.300   Min.   :2.000   Min.   :1.000   Min.   :0.100   setosa    :50
##  1st Qu.:5.100   1st Qu.:2.800   1st Qu.:1.600   1st Qu.:0.300   versicolor:50
##  Median :5.800   Median :3.000   Median :4.350   Median :1.300   virginica :50
##  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
```

```
any(is.na(iris)) #checking null value
```

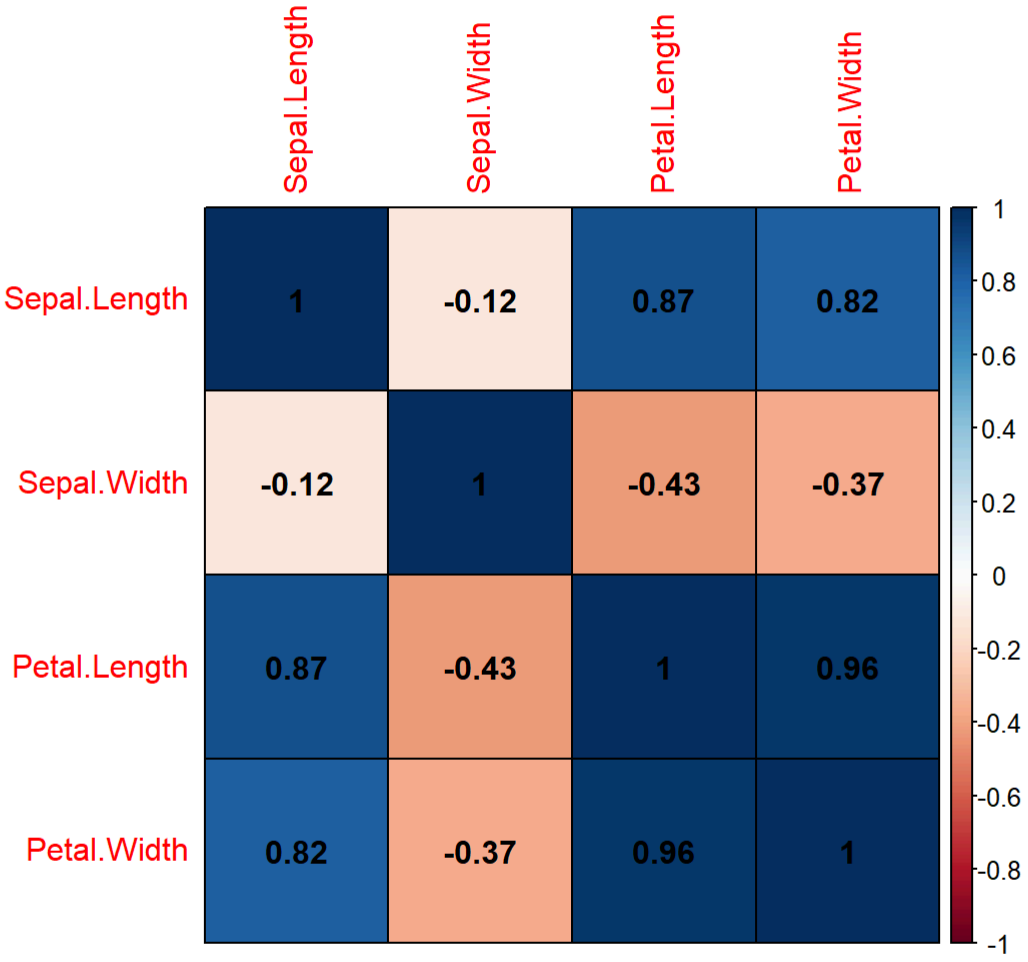
```
## [1] FALSE
```

This dataset contains no missing values, which is an advantage for me.

Correlation Analysis:

Let’s check how the variables in the Iris dataset are correlated with each other.

```
iris_numeric <- iris[, sapply(iris, is.numeric)] #storing numeric columns
corr_coef<-cor(iris_numeric)# correlation coefficient
corrplot(corr_coef,"color",addgrid.col = T,
  addCoef.col = T)
```



Finding correlation coefficients with absolute values greater than or equal to 0.5, while removing **self-correlation**.

```
high_correlation<-which(abs(corr_coef)>=0.5,arr.ind = T)
high_correlation<-high_correlation[high_correlation[,1] != high_correlation[,2],]
result <- data.frame(
  Variable1 = character(),
  Variable2 = character(),
  Correlation = numeric(),
  stringsAsFactors = FALSE
)

for (i in 1:nrow(high_correlation)) {
  row <- high_correlation[i, 1]
  col <- high_correlation[i, 2]
  result <- rbind(result, data.frame(
    Variable1 = colnames(iris_numeric)[row],
    Variable2 = colnames(iris_numeric)[col],
    Correlation = corr_coef[row, col]
  ))
}
result <- result[!duplicated(t(apply(result, 1, sort))), ] # remove duplicate

kable(result, caption = "Variable Pairs with Absolute Correlation Coefficient >= 0.5")
```

Variable Pairs with Absolute Correlation Coefficient >= 0.5

Variable1	Variable2	Correlation

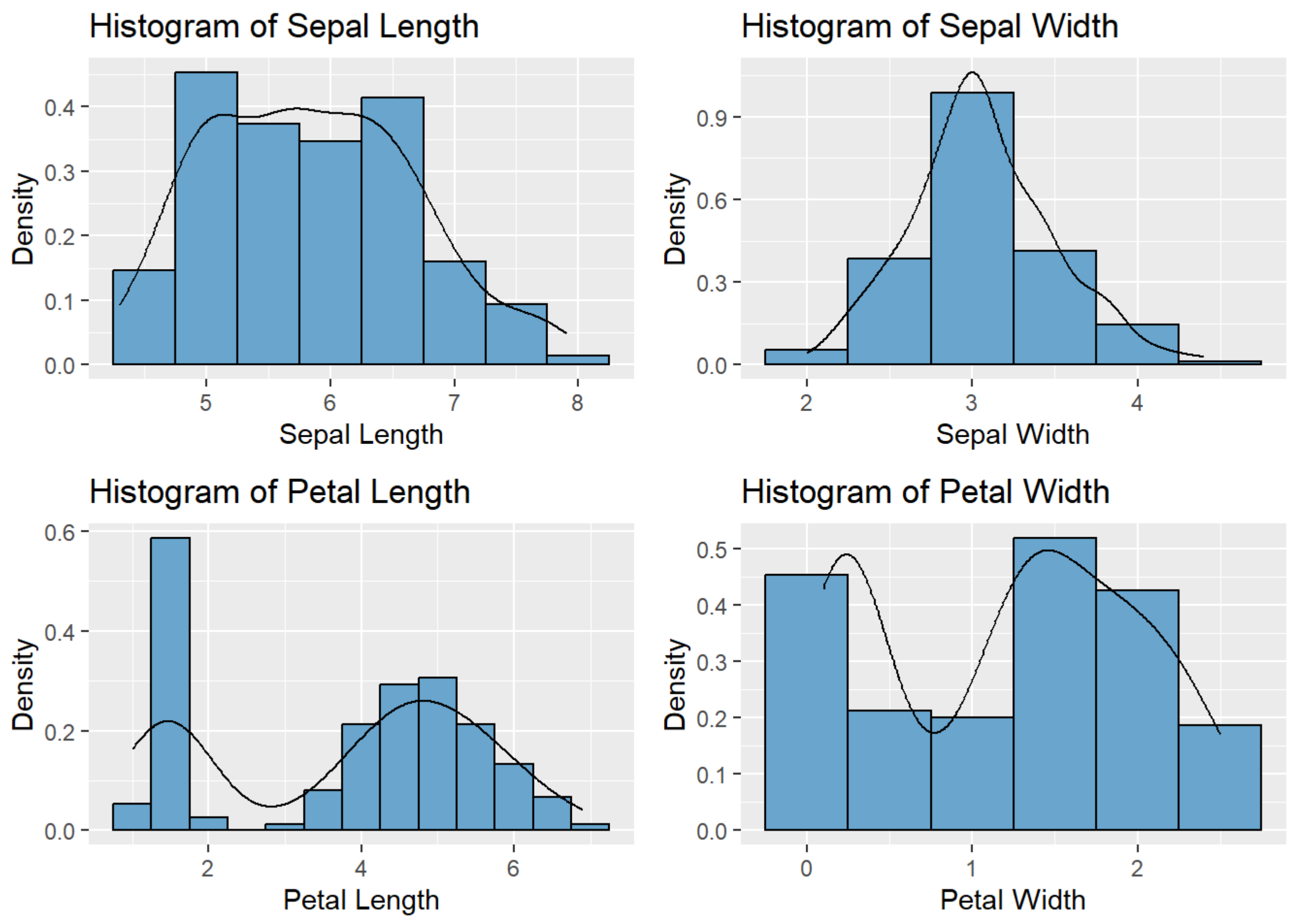
	Variable1	Variable2	Correlation
1	Petal.Length	Sepal.Length	0.8717538
2	Petal.Width	Sepal.Length	0.8179411
4	Petal.Width	Petal.Length	0.9628654

Data Visualisation:

Now, let’s check the data through visualization.

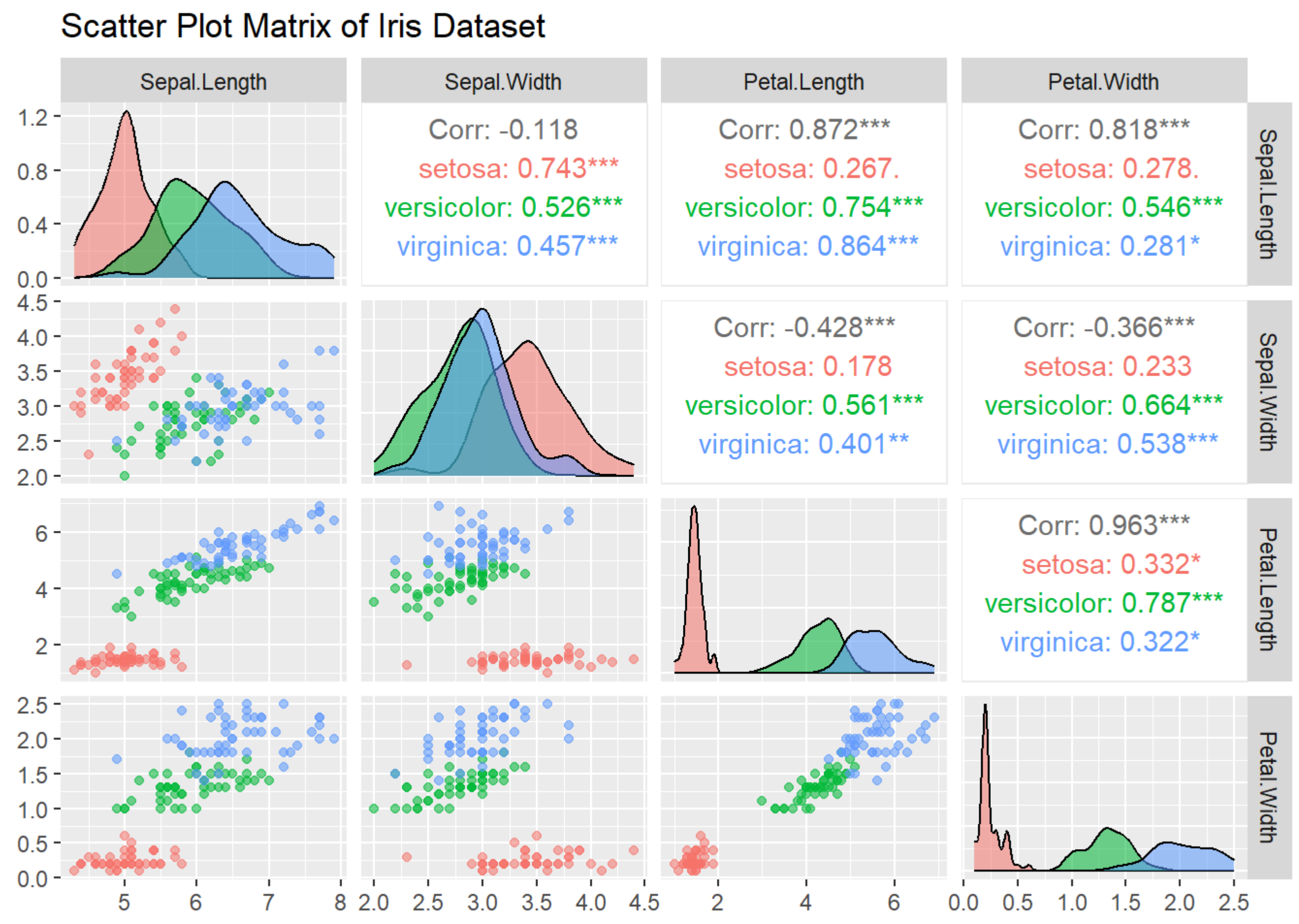
1.Histograms of Iris Dataset Variables:

```
p1 <- ggplot(iris, aes(x = Sepal.Length)) +
  geom_histogram(aes(y = after_stat(density)), binwidth = 0.5, fill = "skyblue3", color = "bl
ack") +
  geom_density(color = "black", linewidth = 0.5, alpha = 0.1) +
  labs(title = "Histogram of Sepal Length",
       x = "Sepal Length",
       y = "Density")

p2 <- ggplot(iris, aes(x = Sepal.Width)) +
  geom_histogram(aes(y = after_stat(density)), binwidth = 0.5, fill = "skyblue3", color = "bl
ack") +
  geom_density(color = "black", size = 0.5, alpha = 0.1) +
  labs(title = "Histogram of Sepal Width",
       x = "Sepal Width",
       y = "Density")

p3 <- ggplot(iris, aes(x = Petal.Length)) +
  geom_histogram(aes(y = after_stat(density)), binwidth = 0.5, fill = "skyblue3", color = "bl
ack") +
  geom_density(color = "black", size = 0.5, alpha = 0.1) +
  labs(title = "Histogram of Petal Length",
       x = "Petal Length",
       y = "Density")

p4 <- ggplot(iris, aes(x = Petal.Width)) +
  geom_histogram(aes(y = after_stat(density)), binwidth = 0.5, fill = "skyblue3", color = "bl
ack") +
  geom_density(color = "black", size = 0.5, alpha = 0.1) +
  labs(title = "Histogram of Petal Width",
       x = "Petal Width",
       y = "Density")

grid.arrange(p1, p2, p3, p4)
```



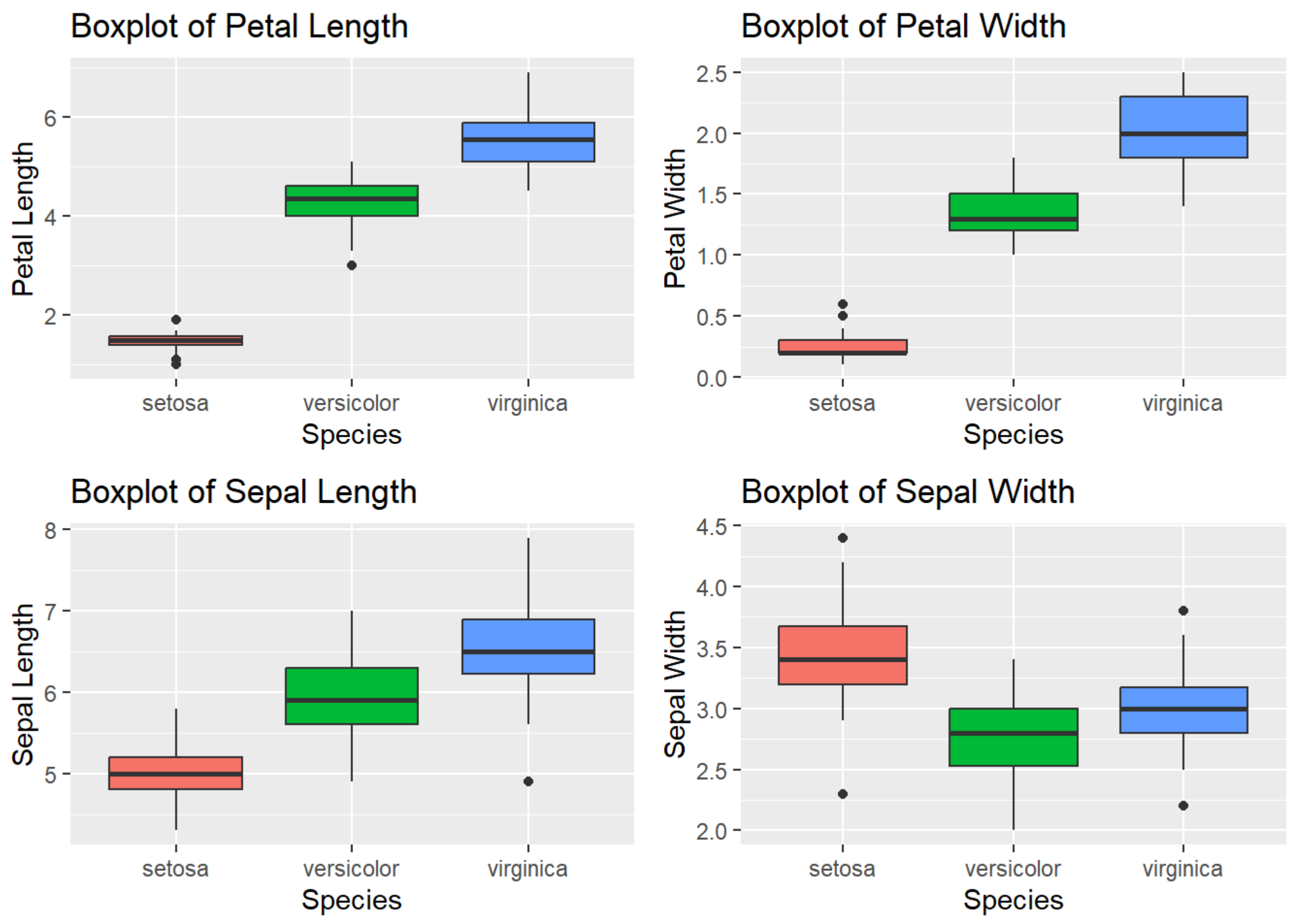
2.Pair Plots of Iris Dataset Variables:

```
ggpairs(iris, aes(color = Species, alpha = 0.8),
        columns = c("Sepal.Length", "Sepal.Width", "Petal.Length", "Petal.Width"),
        title = "Scatter Plot Matrix of Iris Dataset")
```



3.Boxplots of Iris Dataset Variables:

```
b1 <- ggplot(iris, aes(x = Species,y = Petal.Length, fill = Species))+
  geom_boxplot(position = "dodge2")+
  labs(title = "Boxplot of Petal Length",
       x = "Species",
       y = "Petal Length")+
  theme(legend.position = "none")

b2 <- ggplot(iris, aes(x = Species,y = Petal.Width, fill = Species))+
  geom_boxplot(position = "dodge2")+
  labs(title = "Boxplot of Petal Width",
       x = "Species",
       y = "Petal Width")+
  theme(legend.position = "none")

b3 <- ggplot(iris, aes(x = Species,y = Sepal.Length, fill = Species))+
  geom_boxplot(position = "dodge2")+
  labs(title = "Boxplot of Sepal Length",
       x = "Species",
       y = "Sepal Length")+
  theme(legend.position = "none")

b4 <- ggplot(iris, aes(x = Species,y =Sepal.Width, fill = Species))+
  geom_boxplot(position = "dodge2")+
  labs(title = "Boxplot of Sepal Width",
       x = "Species",
       y = "Sepal Width")+
  theme(legend.position = "none")

grid.arrange(b1, b2, b3, b4, nrow = 2, ncol = 2)
```



4.Findings:
* For `petal length` , `petal width` , and `sepal length` , **Setosa** values are lower and **Virginica** values are higher.
* For `sepal width` , **Setosa** values are higher and **Versicolor** values are lower.

Predictive modeling:

Let’s create models to predict species and evaluate its accuracy.

1.Separating the Dataset into Training and Testing Sets:
To ensure a robust evaluation of my model, I use an **80:20 split**, allocating 80% of the data for training and 20% for testing.

```
set.seed(598)
sample_size <- floor(0.8 * nrow(iris))
train_indices <- sample(seq_len(nrow(iris)), size = sample_size)
train_data <- iris[train_indices, ]
test_data <- iris[-train_indices, ]
cat("Training data has", nrow(train_data), "rows and", ncol(train_data), "columns.\n","Testin
g data has", nrow(test_data), "rows and", ncol(test_data), "columns.\n")
```

```
## Training data has 120 rows and 5 columns.
##  Testing data has 30 rows and 5 columns.
```

2.Multinomial Logistic Regression:

The `multinom` function in R, part of the `nnet` package, is used for multinomial logistic regression. I use it as the dependent variable `Species` is categorical with more than two levels(3 levels).

```
model <- multinom(Species ~ Sepal.Length + Sepal.Width + Petal.Length + Petal.Width, data = t
rain_data)
```

```
summary(model)
```

```
## Call:
## multinom(formula = Species ~ Sepal.Length + Sepal.Width + Petal.Length + 
##     Petal.Width, data = train_data)
## 
## Coefficients:
##            (Intercept) Sepal.Length Sepal.Width Petal.Length Petal.Width
## versicolor    41.31307    -5.254853   -15.77527     14.29678   -4.835274
## virginica    -42.27493    -6.309020   -27.11442     26.86508   31.004972
## 
## Std. Errors:
##            (Intercept) Sepal.Length Sepal.Width Petal.Length Petal.Width
## versicolor    78.69511     307.3547    385.7167     159.1265    59.13487
## virginica     86.07169     307.4410    386.1755     160.0024    63.55702
## 
## Residual Deviance: 4.320307 
## AIC: 24.32031
```

```
predictions <- predict(model, test_data)
confusion_matrix <- table(predictions, test_data$Species)
print(confusion_matrix)
```

```
##
## predictions  setosa versicolor virginica
##   setosa         11          0         0
##   versicolor      0          8         1
##   virginica       0          0        10
```

```
accuracy <- sum(diag(confusion_matrix)) / sum(confusion_matrix)
cat("Accuracy of the model:", accuracy * 100, "%\n")
```

```
## Accuracy of the model: 96.66667 %
```

3.Random Forest:
This function `randomForest()` is from the `randomForest` package in R, which implements the Random Forest algorithm. Random Forest is an ensemble learning method that builds multiple decision trees and merges them to get a more accurate and stable prediction.

```
model_rf <- randomForest(Species ~ Sepal.Length + Sepal.Width + Petal.Length + Petal.Width, d
ata = train_data)

print(model_rf)
```

```
##
## Call:
##  randomForest(formula = Species ~ Sepal.Length + Sepal.Width +      Petal.Length + Petal.W
idth, data = train_data)
##                Type of random forest: classification
##                      Number of trees: 500
## No. of variables tried at each split: 2
##
##         OOB estimate of  error rate: 2.5%
## Confusion matrix:
##            setosa versicolor virginica class.error
## setosa         39          0         0  0.00000000
## versicolor      0         41         1  0.02380952
## virginica       0          2        37  0.05128205
```

```
predictions_rf <- predict(model_rf, test_data)

confusion_matrix_rf <- table(predictions_rf, test_data$Species)

print(confusion_matrix_rf)
```

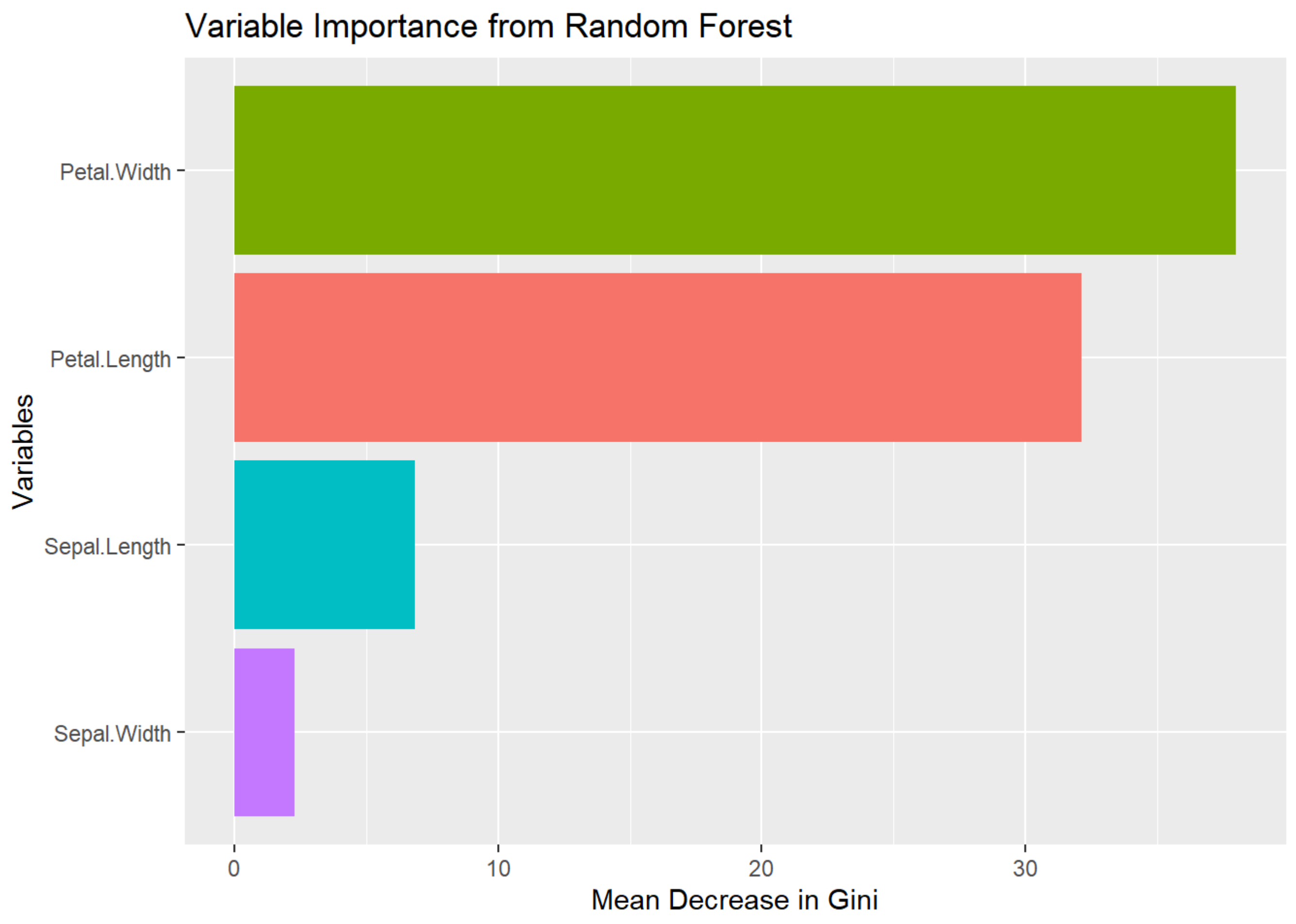
```
##
## predictions_rf setosa versicolor virginica
##     setosa         11          0         0
##     versicolor      0          7         3
##     virginica       0          1         8
```

```
accuracy_rf <- sum(diag(confusion_matrix_rf)) / sum(confusion_matrix_rf)
cat("Accuracy of the model:", accuracy_rf * 100, "%\n")
```

```
## Accuracy of the model: 86.66667 %
```

4.Assessing Variable Importance:

```
importance_df <- data.frame(Variable = rownames(importance(model_rf)),
                            Importance = importance(model_rf)[, "MeanDecreaseGini"])


ggplot(importance_df, aes(x = reorder(Variable, Importance), y = Importance,fill = Variable))
+
  geom_bar(stat = "identity") +
  coord_flip() +
  xlab("Variables") +
  ylab("Mean Decrease in Gini") +
  ggtitle("Variable Importance from Random Forest")+
  theme(legend.position = "none")
```



As a variable, petal measurements are more important than sepal measurements.

5.Findings:

```
e1 <- data.frame(Model = c("Multinomial Logistic Regression","Random Forest"),
                 Accuracy = c(accuracy,accuracy_rf))
kable(e1, caption = "Model vs. Accuracy")
```

Model vs. Accuracy

Model	Accuracy
Multinomial Logistic Regression	0.9666667
Random Forest	0.8666667

It is evident that the **multinomial logistic regression** model provides better accuracy.

Separate Datasets for Sepal and Petal Measurements:

Let’s create separate datasets for sepal and petal measurements from the Iris dataset.

- **Sepal Dataset:** This dataset will contain three columns: `Sepal.Length` , `Sepal.Width` , and `Species.`
- **Petal Dataset:** This dataset will contain three columns: `Petal.Length` , `Petal.Width` , and `Species.`

1.Sepal Dataset Analysis:

```
iris_sepal_train <- train_data %>% select(!(3:4))
iris_sepal_test <- test_data %>% select(!(3:4))
```

```
model_sepal <- multinom(Species ~ ., data = iris_sepal_train)
```

```
summary(model_sepal)
```

```
## Call:
## multinom(formula = Species ~ ., data = iris_sepal_train)
##
## Coefficients:
##            (Intercept) Sepal.Length Sepal.Width
## versicolor   -10.51959     26.48649   -42.24209
## virginica    -24.00165     28.56078   -42.07716
##
## Std. Errors:
##            (Intercept) Sepal.Length Sepal.Width
## versicolor    42.03143     33.07489    43.49537
## virginica     42.07666     33.08517    43.50383
##
## Residual Deviance: 86.63479
## AIC: 98.63479
```

```
predictions_sepal <- predict(model_sepal, iris_sepal_test)
confusion_matrix_sepal <- table(predictions_sepal, iris_sepal_test$Species)
kable(confusion_matrix_sepal,caption =  "Confusion Matrix for Sepal Dataset")
```

Confusion Matrix for Sepal Dataset

	setosa	versicolor	virginica
setosa	10	0	0
versicolor	1	6	2
virginica	0	2	9

```
accuracy_sepal <- sum(diag(confusion_matrix_sepal))/sum(confusion_matrix_sepal)
cat("Accuracy for Sepal dataset: ",accuracy_sepal*100,"%\n")
```

```
## Accuracy for Sepal dataset:  83.33333 %
```

2.Petal Dataset Analysis:

```
iris_petal_train <- train_data %>% select(!(1:2))
iris_petal_test <- test_data %>% select(!(1:2))
```

```
model_petal <- multinom(Species ~ ., data = iris_petal_train)
```

```
summary(model_petal)
```

```
## Call:
## multinom(formula = Species ~ ., data = iris_petal_train)
##
## Coefficients:
##            (Intercept) Petal.Length Petal.Width
## versicolor   -20.93786     5.448841    10.15666
## virginica    -90.16210    14.136092    26.53372
##
## Std. Errors:
##            (Intercept) Petal.Length Petal.Width
## versicolor    44.22288     41.12934    95.14949
## virginica     52.74075     41.36470    95.44721
##
## Residual Deviance: 7.76843
## AIC: 19.76843
```

```
predictions_petal <- predict(model_petal, iris_petal_test)
confusion_matrix_petal <- table(predictions_petal, iris_petal_test$Species)
kable(confusion_matrix_petal, caption = "Confusion Matrix for Petal Dataset")
```

Confusion Matrix for Petal Dataset

	setosa	versicolor	virginica
setosa	11	0	0

	setosa	versicolor	virginica
versicolor	0	6	2
virginica	0	2	9

```
accuracy_petal <- sum(diag(confusion_matrix_petal))/sum(confusion_matrix_petal)
cat("Accuracy for Petal dataset: ", accuracy_petal*100,"%\n")
```

```
## Accuracy for Petal dataset:  86.66667 %
```

3.Findings:

```
e2 <- data.frame(Model = c("For Sepal Dataset","For Petal Dataset"),
                 Accuracy = c(accuracy_sepal,accuracy_petal))
kable(e2, caption = "Model vs. Accuracy")
```

Model vs. Accuracy

Model	Accuracy
For Sepal Dataset	0.8333333
For Petal Dataset	0.8666667

The model for the petal dataset achieves higher accuracy compared to the model for the sepal dataset. This can be justified by examining the variable importance plot, which highlights the greater relevance of petal measurements for accurate species classification.

Principal Component Analysis (PCA):

1.Analysis:

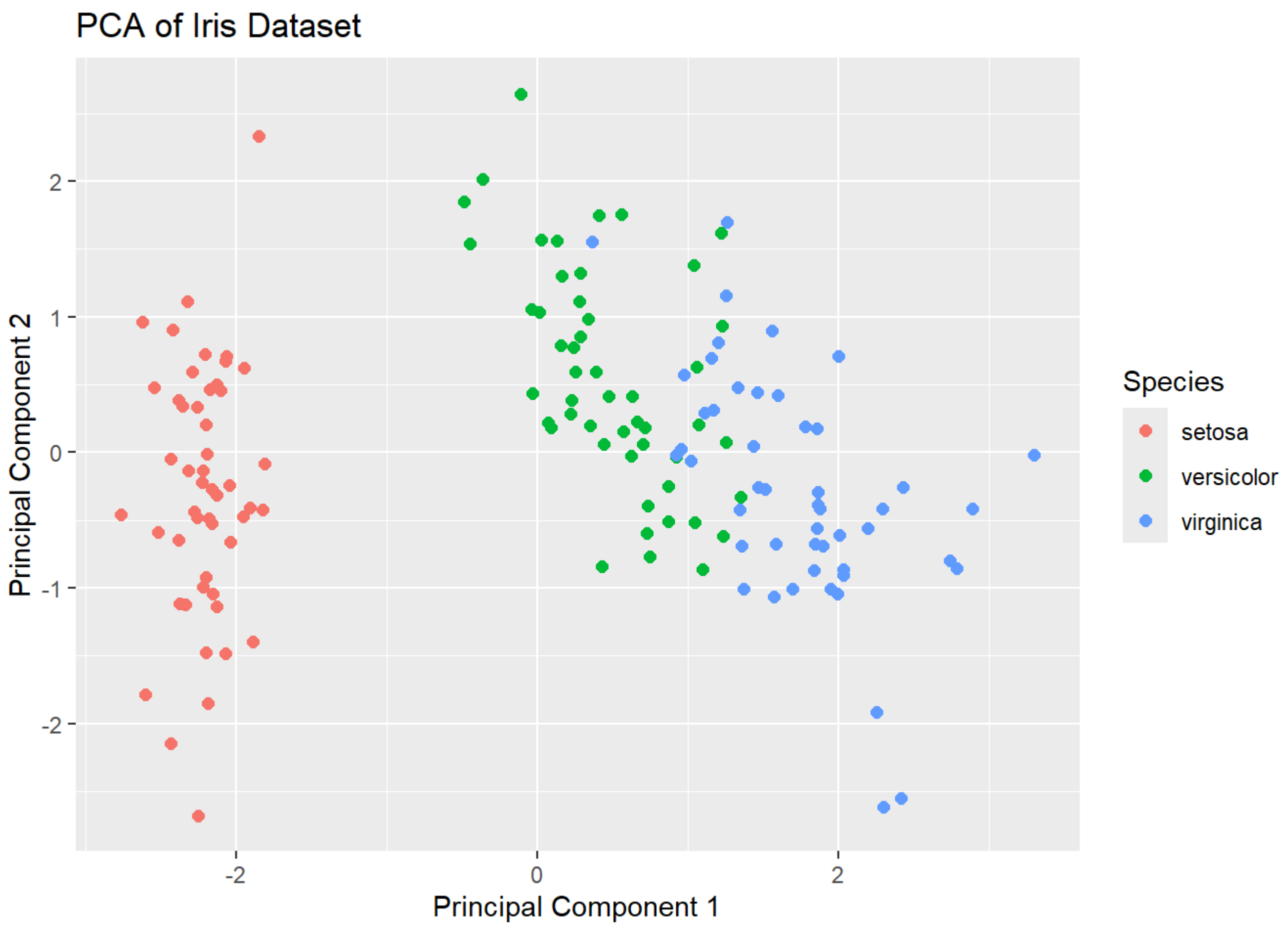
```
iris_data <- iris[, -5]
iris_data_scaled <- scale(iris_data)
pca_result <- prcomp(iris_data_scaled, center = TRUE, scale. = TRUE)
summary(pca_result)
```

```
## Importance of components:
##                           PC1    PC2     PC3     PC4
## Standard deviation     1.7084 0.9560 0.38309 0.14393
## Proportion of Variance 0.7296 0.2285 0.03669 0.00518
## Cumulative Proportion  0.7296 0.9581 0.99482 1.00000
```

```
pca_result$rotation
```

```
##                     PC1         PC2        PC3        PC4
## Sepal.Length  0.5210659 -0.37741762  0.7195664  0.2612863
## Sepal.Width  -0.2693474 -0.92329566 -0.2443818 -0.1235096
## Petal.Length  0.5804131 -0.02449161 -0.1421264 -0.8014492
## Petal.Width   0.5648565 -0.06694199 -0.6342727  0.5235971
```

```
pca_df <- data.frame(pca_result$x, Species = iris$Species)
ggplot(pca_df, aes(PC1, PC2, color = Species)) +
  geom_point(size = 2) +
  ggtitle("PCA of Iris Dataset") +
  xlab("Principal Component 1") +
  ylab("Principal Component 2")
```



2.Findings:
* According to the PCA results, `Setosa` is well separated from the other species, `Versicolor` and `Virginica`.

Separate models for different species:

1.Creating Binary Indicators for Iris Species:
The code provided is used to create new binary indicator columns in the Iris dataset for each species: Setosa, Versicolor, and Virginica. Each new column will indicate whether the species in a particular row matches the specified species or not.

```
iris$setosa <- ifelse(iris$Species == "setosa","setosa","other")
iris$versicolor <- ifelse(iris$Species == "versicolor","versicolor","other")
iris$virginica <- ifelse(iris$Species == "virginica","virginica","other")
```

2.Creating Separate Datasets for Each Iris Species:
Each new dataset includes the measurements of sepals and petals along with a binary indicator column for the specific species.

```
data_setosa <- iris %>%
  select(Sepal.Length, Sepal.Width, Petal.Length, Petal.Width, setosa)

data_versicolor <- iris %>%
  select(Sepal.Length, Sepal.Width, Petal.Length, Petal.Width, versicolor)

data_virginica <- iris %>%
  select(Sepal.Length, Sepal.Width, Petal.Length, Petal.Width, virginica)
```

3.Creating Separate Models for Each Iris Species:

```
model_setosa <- glm(as.factor(setosa) ~ .,data = data_setosa, family = binomial)
```

```
## Warning: glm.fit: algorithm did not converge
```

```
## Warning: glm.fit: fitted probabilities numerically 0 or 1 occurred
```

```
model_versicolor <- glm(as.factor(versicolor) ~ .,
                        data = data_versicolor, family = binomial)

model_virginica <- glm(as.factor(virginica) ~ .,
                       data = data_virginica, family = binomial)
```

```
## Warning: glm.fit: fitted probabilities numerically 0 or 1 occurred
```

4.Summary of Separate Models:

```
summary(model_setosa)
```

```
## 
## Call:
## glm(formula = as.factor(setosa) ~ ., family = binomial, data = data_setosa)
## 
## Coefficients:
##                Estimate Std. Error z value Pr(>|z|)
## (Intercept)     -16.946 457457.097       0        1
## Sepal.Length     11.759 130504.042       0        1
## Sepal.Width       7.842  59415.385       0        1
## Petal.Length    -20.088 107724.594       0        1
## Petal.Width     -21.608 154350.616       0        1
## 
## (Dispersion parameter for binomial family taken to be 1)
## 
##     Null deviance: 1.9095e+02  on 149  degrees of freedom
## Residual deviance: 3.2940e-09  on 145  degrees of freedom
## AIC: 10
## 
## Number of Fisher Scoring iterations: 25
```

```
summary(model_versicolor)
```

```
##
## Call:
## glm(formula = as.factor(versicolor) ~ ., family = binomial, data = data_versicolor)
##
## Coefficients:
##              Estimate Std. Error z value Pr(>|z|)
## (Intercept)    7.3785     2.4993   2.952 0.003155 **
## Sepal.Length  -0.2454     0.6496  -0.378 0.705634
## Sepal.Width   -2.7966     0.7835  -3.569 0.000358 ***
## Petal.Length   1.3136     0.6838   1.921 0.054713 .
## Petal.Width   -2.7783     1.1731  -2.368 0.017868 *
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## (Dispersion parameter for binomial family taken to be 1)
##
##     Null deviance: 190.95  on 149  degrees of freedom
## Residual deviance: 145.07  on 145  degrees of freedom
## AIC: 155.07
##
## Number of Fisher Scoring iterations: 5
```

```
summary(model_virginica)
```

```
##
## Call:
## glm(formula = as.factor(virginica) ~ ., family = binomial, data = data_virginica)
##
## Coefficients:
##              Estimate Std. Error z value Pr(>|z|)
## (Intercept)   -42.638     25.708  -1.659   0.0972 .
## Sepal.Length   -2.465      2.394  -1.030   0.3032
## Sepal.Width    -6.681      4.480  -1.491   0.1359
## Petal.Length    9.429      4.737   1.990   0.0465 *
## Petal.Width    18.286      9.743   1.877   0.0605 .
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## (Dispersion parameter for binomial family taken to be 1)
##
##     Null deviance: 190.954  on 149  degrees of freedom
## Residual deviance:  11.899  on 145  degrees of freedom
## AIC: 21.899
##
## Number of Fisher Scoring iterations: 12
```

> **5.Predicting Iris Species with Multiple Models:**
> I have a dataset of 10 random rows of iris flower measurements. Each row contained the sepal length, sepal width, petal length, petal width, and the actual species of the iris flower. Our goal was to predict the species of these flowers using previously trained models for Setosa, Versicolor, and Virginica.

```
random_rows <- data.frame(
  Sepal.Length = c(5.0, 6.7, 7.3, 5.4, 6.1, 4.9, 6.8, 7.7, 5.5, 6.0),
  Sepal.Width = c(3.4, 3.1, 3.2, 3.9, 2.8, 3.6, 2.8, 2.6, 4.2, 3.0),
  Petal.Length = c(1.5, 4.7, 6.3, 1.3, 4.0, 1.4, 4.8, 6.9, 1.5, 4.5),
  Petal.Width = c(0.2, 1.5, 2.4, 0.4, 1.3, 0.1, 1.4, 2.3, 0.2, 1.5),
  Actual_Species = c('setosa', 'versicolor', 'virginica', 'setosa', 'versicolor', 'setosa',
'versicolor', 'virginica', 'setosa', 'versicolor')
)

predictions_setosa <- predict(model_setosa, random_rows, type = "response")
predictions_versicolor <- predict(model_versicolor, random_rows, type = "response")
predictions_virginica <- predict(model_virginica, random_rows, type = "response")

combined_predictions <- data.frame(predictions_setosa,predictions_versicolor
                         , predictions_virginica)

final_predictions <- apply(combined_predictions, 1, function(row) {
  colnames(combined_predictions)[which.max(row)]
})

final_results <- random_rows %>%
  mutate(Predicted_Species = final_predictions)

kable(final_results)
```

Sepal.Length	Sepal.Width	Petal.Length	Petal.Width	Actual_Species	Predicted_Species
5.0	3.4	1.5	0.2	setosa	predictions_setosa
6.7	3.1	4.7	1.5	versicolor	predictions_versicolor
7.3	3.2	6.3	2.4	virginica	predictions_virginica
5.4	3.9	1.3	0.4	setosa	predictions_setosa
6.1	2.8	4.0	1.3	versicolor	predictions_versicolor
4.9	3.6	1.4	0.1	setosa	predictions_setosa
6.8	2.8	4.8	1.4	versicolor	predictions_versicolor
7.7	2.6	6.9	2.3	virginica	predictions_virginica
5.5	4.2	1.5	0.2	setosa	predictions_setosa
6.0	3.0	4.5	1.5	versicolor	predictions_versicolor

6.Findings:
After careful scrutiny, I can confidently say that my model perfectly predicts the species.

Feature Engineering:

1.Adding Petal Ratio:
First, we calculated the ratio of petal length to petal width for each iris flower. This new measure, called `petal_ratio` , is created by dividing the `Petal.Length` by the `Petal.Width.` This ratio helps us understand the proportion of the petal's length relative to its width, providing a new perspective on the flower's morphology.

```
iris$petal_ratio <- iris$Petal.Length/iris$Petal.Width
```

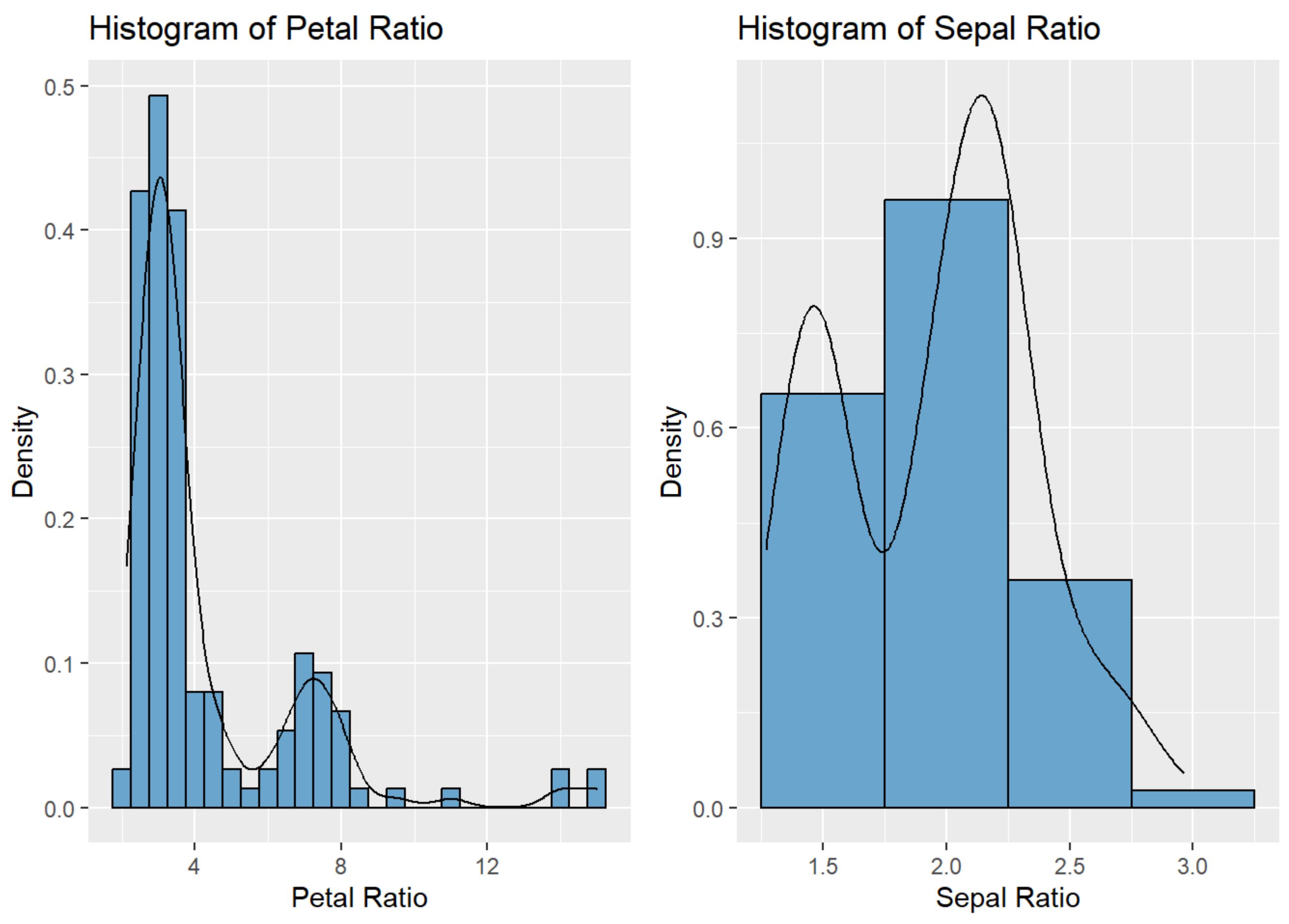
2.Adding Sepal Ratio:
Repeat the same process for sepal measurements. This ratio offers insight into the proportion of the sepal's length compared to its width, adding another dimension to our understanding of the flower's structure.

```
iris$sepal_ratio <- iris$Sepal.Length/iris$Sepal.Width
iris_new <- iris %>%
  select(petal_ratio,sepal_ratio,Species)
```

3.Exploring the Enhanced Iris Dataset:
After creating two new variables— `petal_ratio` and `sepal_ratio` —I store these enhancements in a new dataset named `iris_new.` This updated dataset now includes additional features that offer deeper insights into the morphology of the iris flowers.
To understand these new features better, I embark on a data journey through visualization.

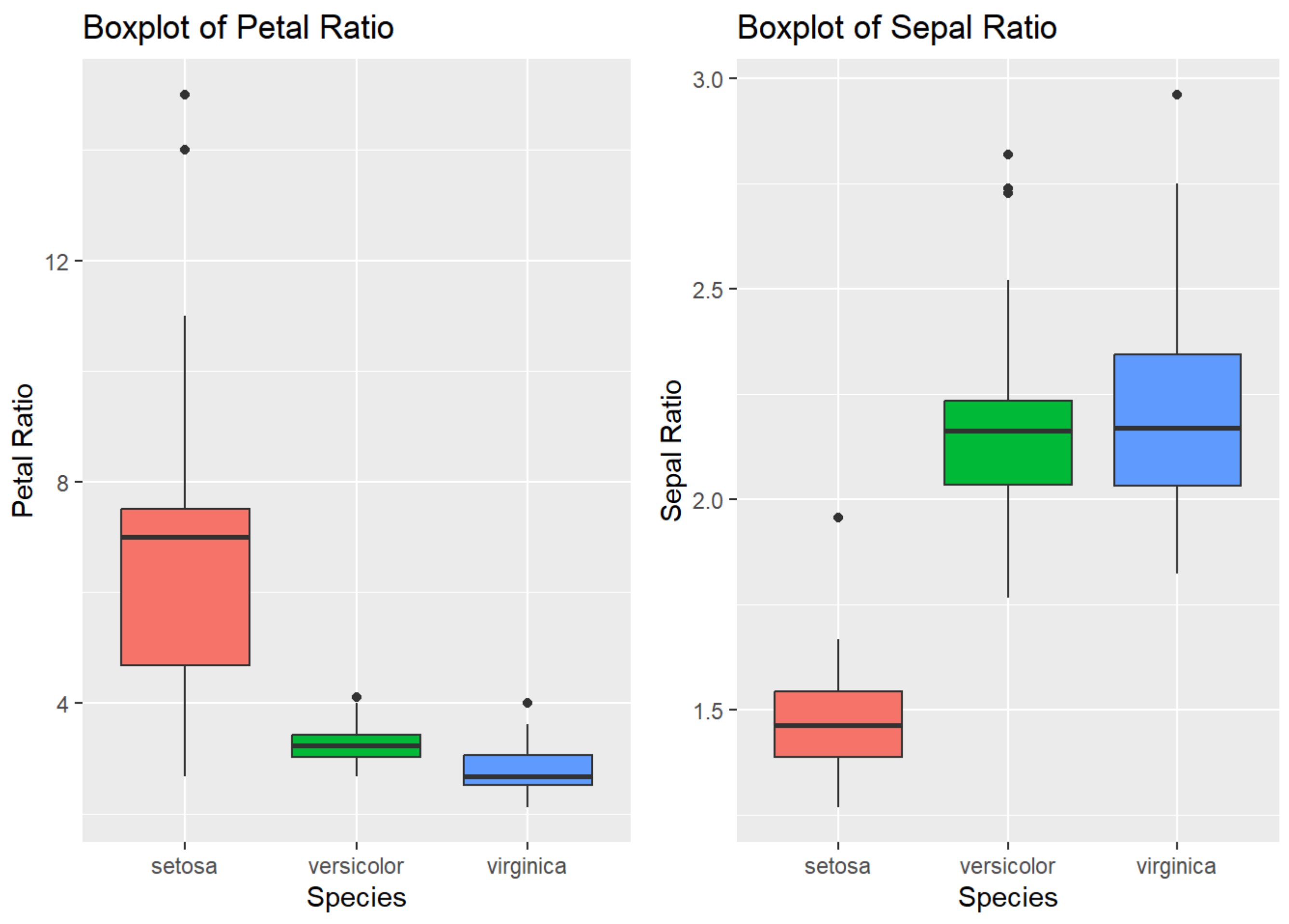
4.Histograms of Enhanced Iris Dataset Variables:

```
d1 <- ggplot(iris_new, aes(x = petal_ratio)) +
  geom_histogram(aes(y = after_stat(density)), binwidth = 0.5, fill = "skyblue3", color = "bl
ack") +
  geom_density(color = "black", size = 0.5, alpha = 0.1) +
  labs(title = "Histogram of Petal Ratio",
       x = "Petal Ratio",
       y = "Density")

d2 <- ggplot(iris_new, aes(x = sepal_ratio)) +
  geom_histogram(aes(y = after_stat(density)), binwidth = 0.5, fill = "skyblue3", color = "bl
ack") +
  geom_density(color = "black", size = 0.5, alpha = 0.1) +
  labs(title = "Histogram of Sepal Ratio",
       x = "Sepal Ratio",
       y = "Density")

grid.arrange(d1,d2,ncol=2)
```



5.Boxplots of Enhanced Iris Dataset Variables:

```
c1 <- ggplot(iris_new, aes(x = Species,y = petal_ratio, fill = Species))+
  geom_boxplot(position = "dodge2")+
  labs(title = "Boxplot of Petal Ratio",
       x = "Species",
       y = "Petal Ratio")+
  theme(legend.position = "none")

c2 <- ggplot(iris_new, aes(x = Species,y = sepal_ratio, fill = Species))+
  geom_boxplot(position = "dodge2")+
  labs(title = "Boxplot of Sepal Ratio",
       x = "Species",
       y = "Sepal Ratio")+
  theme(legend.position = "none")

grid.arrange(c1, c2, ncol = 2)
```



6.Predictive Modeling for Enhanced Iris Dataset:

```
set.seed(895)
trainIndex <- createDataPartition(iris_new$Species, p = 0.8,
                                  list = FALSE,
                                  times = 1)

iris_new_train <- iris_new[trainIndex, ]
iris_new_test <- iris_new[-trainIndex, ]

cat("Training data has", nrow(iris_new_train), "rows and", ncol(iris_new_train), "column
s.\n")
```

```
## Training data has 120 rows and 3 columns.
```

```
cat("Testing data has", nrow(iris_new_test), "rows and", ncol(iris_new_test), "columns.\n")
```

```
## Testing data has 30 rows and 3 columns.
```

7.Multinomial Logistic Regression for Enhanced Iris Dataset:

```
model_iris_new <- multinom(as.factor(Species) ~ ., data = iris_new_train)
```

```
summary(model_iris_new)
```

```
## Call:
## multinom(formula = as.factor(Species) ~ ., data = iris_new_train)
##
## Coefficients:
##            (Intercept) petal_ratio sepal_ratio
## versicolor   -21.98796   -11.09885    34.47896
## virginica    -17.71981   -15.71240    38.87406
##
## Std. Errors:
##            (Intercept) petal_ratio sepal_ratio
## versicolor    10.33105    36.27487    62.67663
## virginica     10.33020    36.28734    62.69154
##
## Residual Deviance: 75.79496
## AIC: 87.79496
```

```
predictions_iris_new <- predict(model_iris_new, iris_new_test)
confusion_matrix_iris_new <- table(predictions_iris_new, iris_new_test$Species)
kable(confusion_matrix_iris_new, caption = "Confusion Matrix for New Iris Dataset")
```

Confusion Matrix for New Iris Dataset

	setosa	versicolor	virginica

	setosa	versicolor	virginica
setosa	10	0	0
versicolor	0	10	2
virginica	0	0	8

```
accuracy_new_iris <- sum(diag(confusion_matrix_iris_new))/sum(confusion_matrix_iris_new)
cat("Accuracy for New Iris dataset: ", accuracy_new_iris*100,"%\n")
```

```
## Accuracy for New Iris dataset:  93.33333 %
```

8.Random Forest for Enhanced Iris Dataset:

```
model_new_rf <- randomForest(Species ~ ., data = iris_new_train)
print(model_new_rf)
```

```
##
## Call:
##  randomForest(formula = Species ~ ., data = iris_new_train)
##                Type of random forest: classification
##                      Number of trees: 500
## No. of variables tried at each split: 1
##
##         OOB estimate of  error rate: 20.83%
## Confusion matrix:
##            setosa versicolor virginica class.error
## setosa         40          0         0       0.000
## versicolor      0         29        11       0.275
## virginica       0         14        26       0.350
```

```
predictions_new_rf <- predict(model_new_rf, iris_new_test)
confusion_matrix_new_rf <- table(predictions_new_rf, iris_new_test$Species)
print(confusion_matrix_new_rf)
```

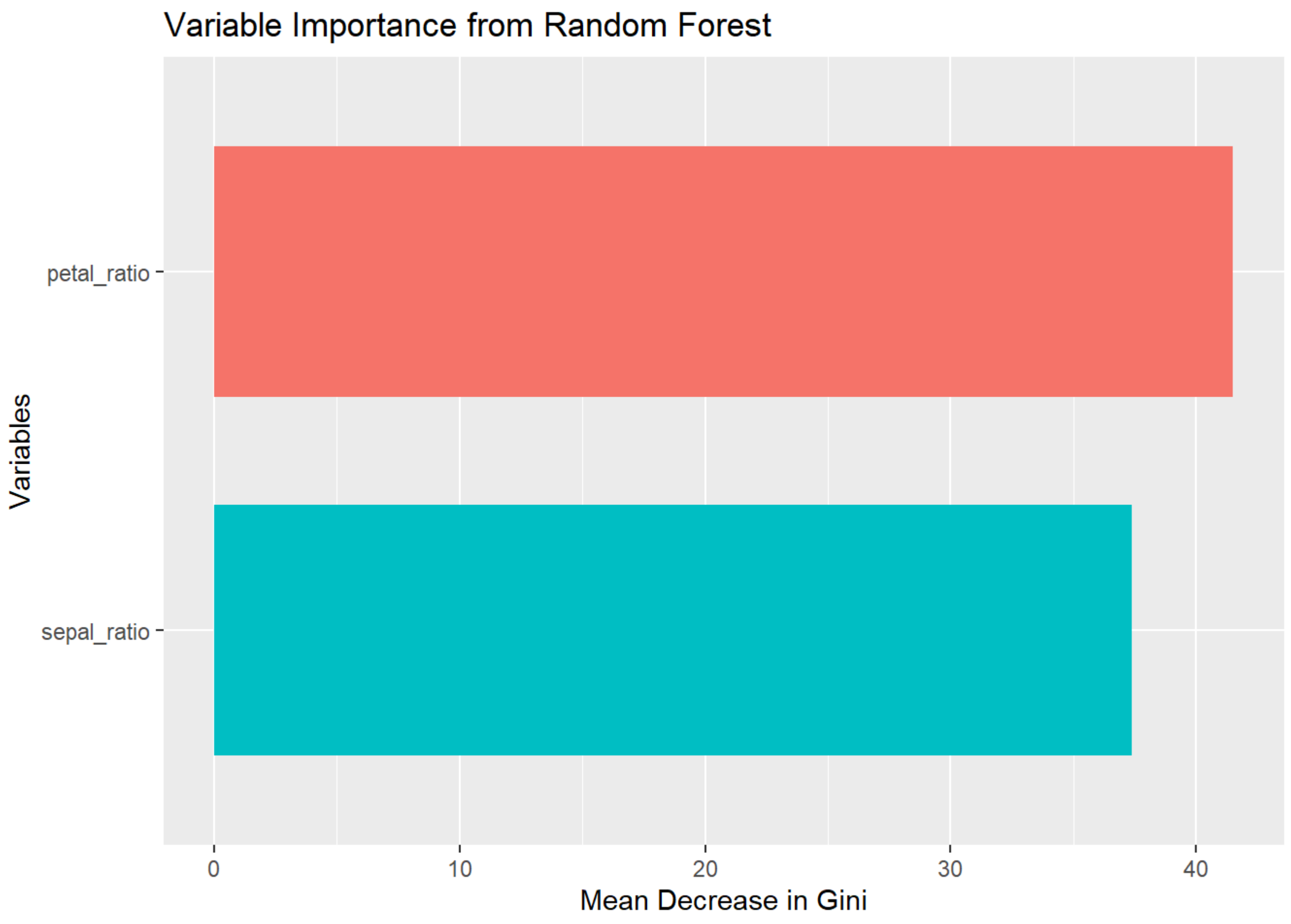
```
##
## predictions_new_rf setosa versicolor virginica
##         setosa          9          0         0
##         versicolor      1          9         1
##         virginica       0          1         9
```

```
accuracy_new_rf <- sum(diag(confusion_matrix_new_rf))/sum(confusion_matrix_new_rf)
cat("Accuracy for New Iris dataset: ", accuracy_new_rf*100,"%\n")
```

```
## Accuracy for New Iris dataset:  90 %
```

9.Assessing Variable Importance through Random Forest:

```r
importance_new_df <- data.frame(Variable = rownames(importance(model_new_rf)),
                            Importance = importance(model_new_rf)[, "MeanDecreaseGini"])


ggplot(importance_new_df, aes(x = reorder(Variable, Importance), y = Importance,fill = Variab
le)) +
  geom_bar(stat = "identity",width = 0.7) +
  coord_flip() +
  xlab("Variables") +
  ylab("Mean Decrease in Gini") +
  ggtitle("Variable Importance from Random Forest")+
  theme(legend.position = "none")
```



As a variable, `petal_ratio` is more important than `sepal_ratio`.

10.Findings:

```
e3 <- data.frame(Model = c("Multinomial Logistic Regression","Random Forest"),
                 Accuracy = c(accuracy_new_iris,accuracy_new_rf))
kable(e3, caption = "Model for Enhanced Iris Dataset  vs. Accuracy")
```

Model for Enhanced Iris Dataset vs. Accuracy

Model	Accuracy
Multinomial Logistic Regression	0.9333333
Random Forest	0.9000000

Here in case of enhanced Iris Dataset, multinomial logistic regression also demonstrates better accuracy compared to the random forest model.

11.Create an Interactive Form for Predicting Species:

For this purpose, I use the model created by **multinomial logistic regression**. By applying **multinomial logistic regression**, I leverage its ability to classify observations into more than two categories, which enhances the **accuracy** and **reliability** of the predictions.

```r
ui <- fluidPage(
  titlePanel("Iris Species Predictor"),
  sidebarLayout(
    sidebarPanel(
      numericInput("petal_ratio", "Petal Ratio:", value = "null"),
      numericInput("sepal_ratio", "Sepal Ratio:", value = "null"),
      actionButton("predict", "Predict")
    ),
    mainPanel(
      textOutput("species"),
      tableOutput("probabilities")
    )
  )
)

server <- function(input, output) {
  observeEvent(input$predict, {
    new_data <- data.frame(
      petal_ratio = input$petal_ratio,
      sepal_ratio = input$sepal_ratio
    )

    prediction <- predict(model_iris_new, new_data, type = "prob")
    species <- predict(model_iris_new, new_data)

    table <- data.frame(Species = c("Setosa","Versicolor","Virginica"),
                        Probability = prediction)

    output$species <- renderText({
      paste("Predicted Species:", species)
    })

    output$probabilities <- renderTable({
      table
    })
  })
}

shinyApp(ui = ui, server = server)
```

Iris Species Predictor

Petal Ratio:

Sepal Ratio:

Predict

> In the output table, the species-wise probabilities are provided. The species with the highest probability is the predicted species.

The End