INTRODUCTION TO MACHINE LEARNING (IRIS DATASET)

HANDS-ON WORKSHOP 'BioAI for Research'

January 2025

Presented by

Gitanjali Yadav (NIPGR | University of Cambridge) and Anudev S

OUTLINE

R Studio

What is R and R markdown

What is Iris dataset?

Machine learning overview

Decision tree

Logistic Regression

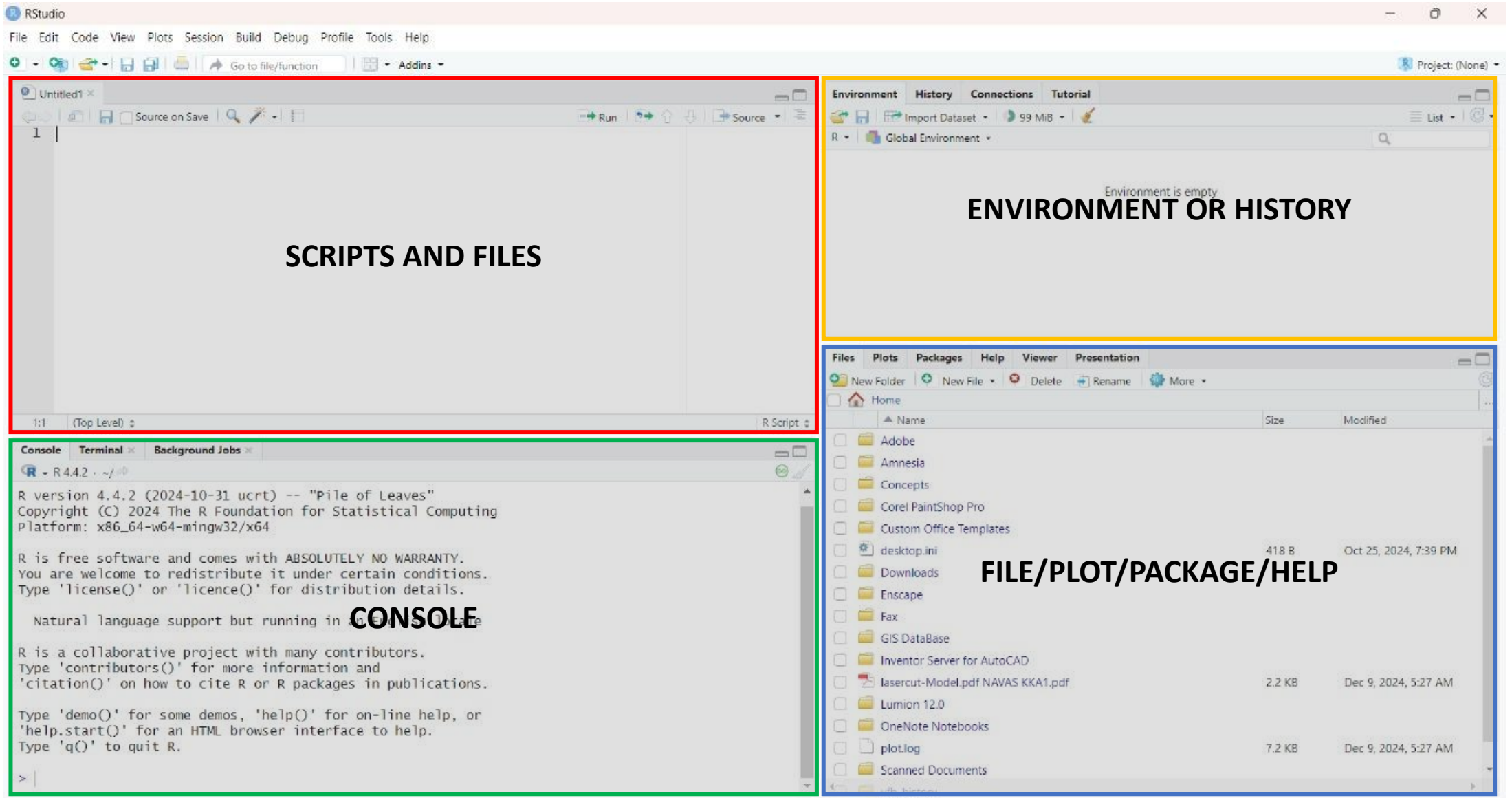
What is R and R markdown

- R is a free, open-source programming language and statistical software used for data analysis and visualization.
- It offers an extensive ecosystem of packages, which are libraries of pre-built functions that simplify complex tasks such as data manipulation (e.g., dplyr), visualization (e.g., ggplot2), and machine learning (e.g., caret).
- R Markdown is an essential tool in R that allows users to create dynamic, reproducible documents by combining code, results, and narrative in a single file.
- With R Markdown, you can easily generate reports, presentations, or web pages, making it an ideal choice for data-driven projects that need seamless documentation and collaboration.

We will use an R Markdown file in RStudio to demonstrate Machine Learning with the famous Iris dataset.

Let’s explore the R Studio and R markdown file.

R Studio


RStudio
File Edit Code View Plots Session Build Debug Profile Tools Help
Go to file/function
Addins
Project: (None)
Untitled1
Source on Save
Run
Source
1
1:1
(Top Level)
R Script
SCRIPTS AND FILES
Environment History Connections Tutorial
Import Dataset
99 MiB
List
R
Global Environment
Environment is empty
ENVIRONMENT OR HISTORY
Console Terminal Background Jobs
R 4.4.2 · ~/
R version 4.4.2 (2024-10-31 ucrt) -- "Pile of Leaves"
Copyright (C) 2024 The R Foundation for Statistical Computing
Platform: x86_64-w64-mingw32/x64
R is free software and comes with ABSOLUTELY NO WARRANTY.
You are welcome to redistribute it under certain conditions.
Type 'license()' or 'licence()' for distribution details.
Natural language support but running in
CONSOLE
R is a collaborative project with many contributors.
Type 'contributors()' for more information and
'citation()' on how to cite R or R packages in publications.
Type 'demo()' for some demos, 'help()' for on-line help, or
'help.start()' for an HTML browser interface to help.
Type 'q()' to quit R.
>
Files Plots Packages Help Viewer Presentation
New Folder
New File
Delete
Rename
More
Home
Name
Size
Modified
Adobe
Amnesia
Concepts
Corel PaintShop Pro
Custom Office Templates
desktop.ini 418 B Oct 25, 2024, 7:39 PM
Downloads
Enscape
Fax
GIS DataBase
Inventor Server for AutoCAD
lasercut-Model.pdf NAVAS KKA1.pdf 2.2 KB Dec 9, 2024, 5:27 AM
Lumion 12.0
OneNote Notebooks
plot.log 7.2 KB Dec 9, 2024, 5:27 AM
Scanned Documents
FILE/PLOT/PACKAGE/HELP

R Studio

SCRIPTS AND FILES

These are sections where you write, save, and organize your R code for reproducibility. Scripts allow you to execute chunks of code or the entire file in one go.

CONSOLE

This is the interactive area where code is executed immediately after being entered. It's useful for testing small snippets of code or for quick computations.

ENVIRONMENT OR HISTORY

The Environment tab displays active variables, datasets, and functions loaded into memory. The History tab keeps track of previously executed commands, allowing you to revisit or reuse them.

FILE/PLOT/PACKAGE/ HELP

Files: Lets you navigate your working directory and manage files. **Plots:** Displays graphs or charts generated by your code. **Packages:** Helps you install, load, or update R packages. **Help:** Provides access to R documentation and guides for functions or packages.

What is Iris dataset?

The Iris dataset is a collection of measurements of iris flowers that's used to test algorithms and classify species. It was introduced by the British biologist and statistician Ronald Fisher in 1936 as an example of discriminant analysis.

The features include:

- Sepal width
- Sepal length
- Petal width
- Petal length

The dataset includes 150 samples of three 50 iris species:

- Iris setosa
- Iris versicolor
- Iris verginica

Iris setosa

Iris virginica

Iris versicolor

Figure 1. Images of Iris flower species which is present in iris dataset

Source: https://en.wikipedia.org/wiki/Iris_flower_data_set

Machine learning overview

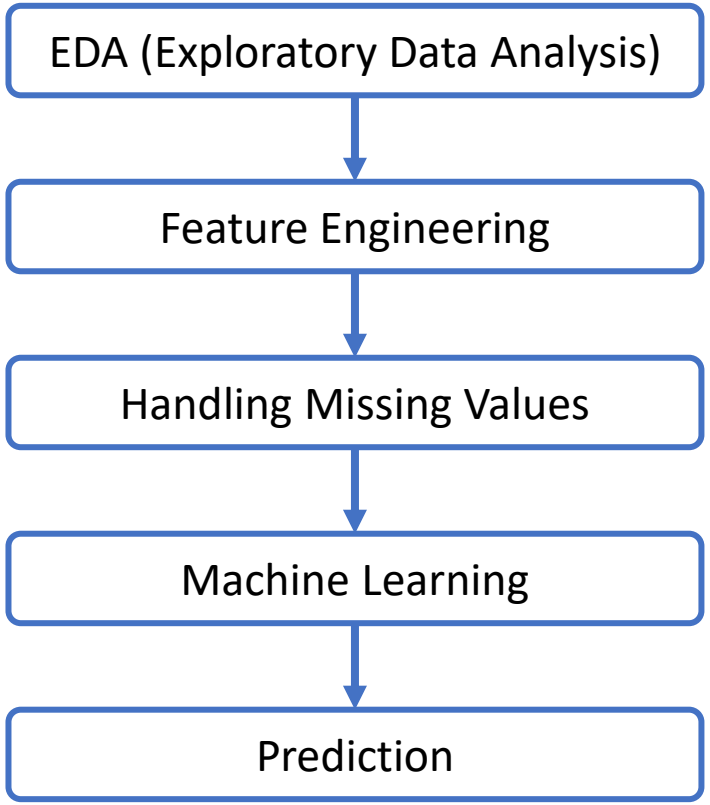
What is machine learning?

Machine learning (ML) is a subset of artificial intelligence (AI) that allows computers to learn and improve from data without being explicitly programmed.

Here we are going to use two machine learning models:

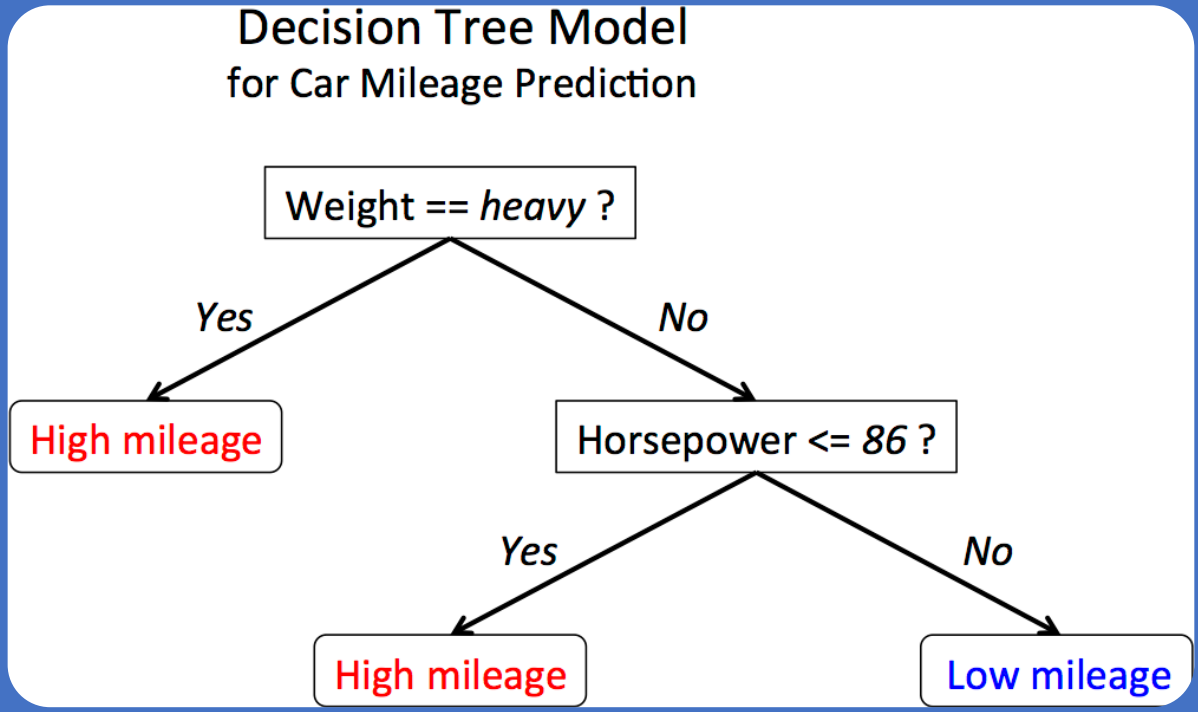
- Decision tree model
- Logistic regression model

Creating machine learning models usually involves the following processes



Decision tree

A decision tree is a machine learning algorithm that uses a tree-like structure to classify or predict data. It's a type of supervised learning algorithm that uses a training data set to learn how to make decisions.



Logistic Regression

What is logistic regression?

Logistic regression is a simple yet powerful machine learning algorithm used for classification tasks. If you're new to machine learning, think of classification as sorting items into categories.

Logistic regression helps us answer questions like:

- Will it rain tomorrow? (Yes or No)
- Is this email spam or not spam? (Spam or Not Spam)
- Is this tumor malignant or benign? (Malignant or Benign)

Let's go through an example

Hours studied	Passed exam?
1	No (0)
2	No (0)
3	Yes (1)
4	Yes (1)
5	Yes (1)

Make predictions:

- A student studies for 3.5 hours. Logistic regression calculates a probability of, say, 0.85 (85% chance of passing).
- If we set a threshold of 0.5 (50%), the model predicts the student will pass (since 0.85 > 0.5).

LET'S GET STARTED

Login to the virtual machine using the link. ***user id*** *and* ***password are*** *provided*

R Packages

A quick recap: An R package is a collection of code, data, and documentation that extends the functionality of the R statistical programming language.

Getting the environment ready:

Chunk 1 (# Install necessary packages)

```
install.packages("ggplot2")
install.packages("psych")
install.packages("plotly")
install.packages("rattle")
install.packages("caret")
```

Chunk 2 (# Load the library)

```
library(ggplot2) # Data visualization
library(psych) # Used for correlation visualizations
library(plotly) # Interactive data visualization
library(rattle) # Graphing decision trees
library(caret) # Machine learning
```

Load your data

Chunk 3

```
data("iris")
```

```
head(iris, n=15) # it gives first 15 rows
```

Output

Sl. No.	Sepal Length	Sepal Width	Petal Length	Petal Width	Species
1	5.1	3.5	1.4	0.2	setosa
2	4.9	3	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	3.6	1.4	0.2	setosa
6	5.4	3.9	1.7	0.4	setosa
7	4.6	3.4	1.4	0.3	setosa
8	5	3.4	1.5	0.2	setosa
9	4.4	2.9	1.4	0.2	setosa
10	4.9	3.1	1.5	0.1	setosa

Load your data

What are there in the table? Let's look at summary of our dataset

Chunk 4

```
summary(iris)
```

Output

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

setosa :50
versicolor:50
virginica :50

Iris setosa

Iris versicolor

Iris virginica

Exploratory data analysis (EDA)

What is EDA?!

Exploratory Data Analysis (EDA) is an important first step in data science projects. It involves looking at and visualizing data to understand its main features, find patterns, and discover how different parts of the data are connected.

Here we are going to explore **Correlation matrix**, **Sepal width**, **Sepal length**, **Petal width** and **Petal length**.

Correlation matrix

A correlation matrix is a rectangular array that shows the correlation coefficients between all possible pairs of variables in a data set. Each cell in the matrix represents the correlation between two variables.

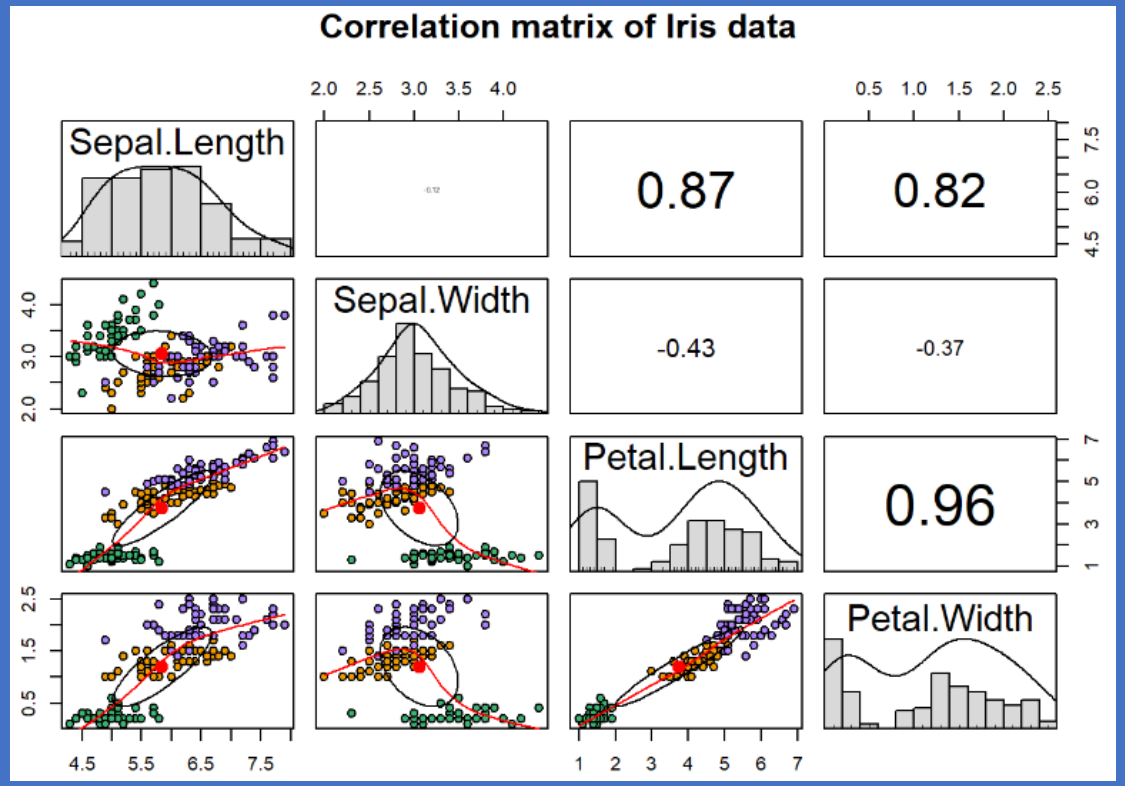
Let's create a correlation matrix using the `pairs.panels()` function from the PSYCH package to see how our variables correlate.

Chunk 5

```
pairs.panels(
  iris[,1:4], # Our data
  scale = TRUE, # Changes the size of the correlation value labels based on
strength
  hist.col = 'grey85’, # Histogram color
  bg = c("mediumseagreen","orange2","mediumpurple1")[iris$Species], # Colors of
the Species levels.
       pch = 21, # The plot characters shape and size.
       main = 'Correlation matrix of Iris data’) # Title.
```

Correlation matrix

Output



What is in this matrix?

- Here the diagonal boxes represents the histogram of the individual features.
- The lower left section represents the scatter plots where each point represents a species of iris flowers, visualizing the relationship between two features.
- The upper right part section represents correlation coeffcient (from -1 to 1) between features.

DATA VISUALIZATION IN R

Visualizing data in R using ggplot2

What is ggplot?

- The ggplot2 package in R is a tool for creating data visualizations, such as plots, charts, and graphs. It's based on the Grammar of Graphics, which allows users to create custom graphics by combining independent components.

```
# It contains three elements
ggplot (data = <DATA>, mapping = aes(<MAPPINGS>)) + <GEOM_FUNCTION>()
```

- The function takes TWO arguments:

  data: Data frame with variables to plot (columns and rows)
  mapping: Aesthetics (How variables are mapped to visual properties of the plot)

- Add a geom_function as a layer (By using +)

  geom_function specifies the type of plot would you like to plot

Visualizing data in R using ggplot2

Data: As the foundation of every graphic, ggplot2 uses data to construct a plot.

Mapping: The mapping of a plot is a set of instructions on how parts of the data are mapped onto aesthetic attributes of geometric objects.

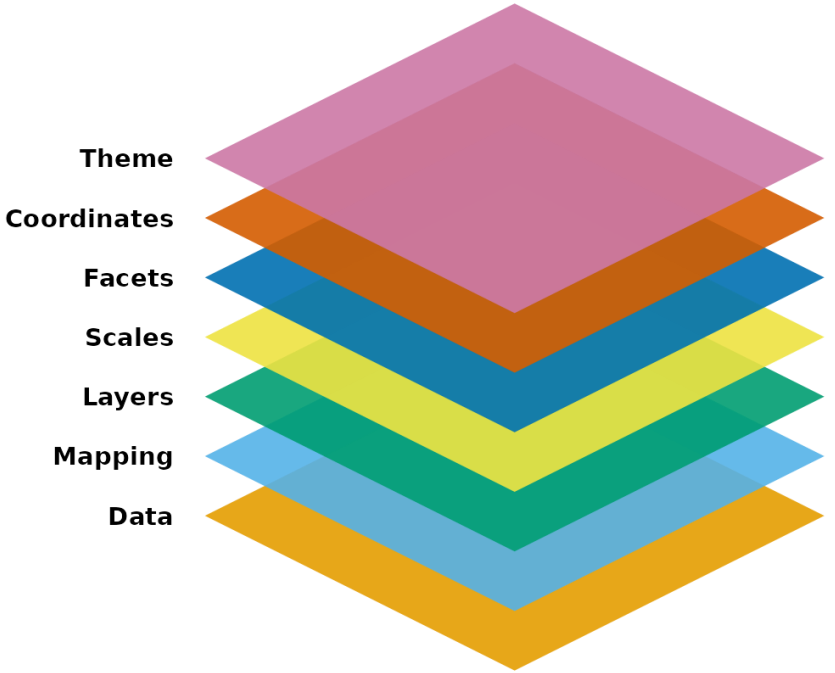
Layers: They take the mapped data and display it in something humans can understand as a representation of the data.

Scales: Scales are important for translating what is shown on the graph back to an understanding of the data.

Facets: Facets can be used to separate small multiples, or different subsets of the data.

Coordinates: You can view the coordinates part of the plot as an interpreter of position aesthetics.

Theme: The theme system controls almost any visuals of the plot that are not controlled by the data and is therefore important for the look and feel of the plot.



Source: https://ggplot2.tidyverse.org/articles/ggplot2.html

Now let's try this on iris data

Sepal width and Sepal length

Let’s go through the box plot of Sepal width and Sepal length

Chunk 6 (# Box plot of Sepal width)

```
ggplot(data = iris, # Load the data
       mapping = aes(x = Species, y = Sepal.Width, #Specify X and Y variables
                     fill = Species)) + # 'fill' color separates our Species levels
       geom_boxplot() + # Species that we want a box plots
       scale_fill_brewer(palette = 'Dark2') + # Change the color of the box plot
       theme_light() + # set light theme
       labs(title = 'Box plot of Sepal width for each species' ,
              x = 'Species' , y = 'Sepal width') # Assign a title, axes name
```

Chunk 7 (# Box plot of Sepal length)

```
ggplot(data = iris,
       mapping = aes(x = Species, y = Sepal.Length, fill = Species)) +
       geom_boxplot() +
       scale_fill_brewer(palette = 'Dark2') +
       theme_light() +
       labs(title = 'Box plot of Sepal length for each species',
            x = 'Species', y = 'Sepal length')
```

Sepal width and Sepal length

Sepal width

Maximum value

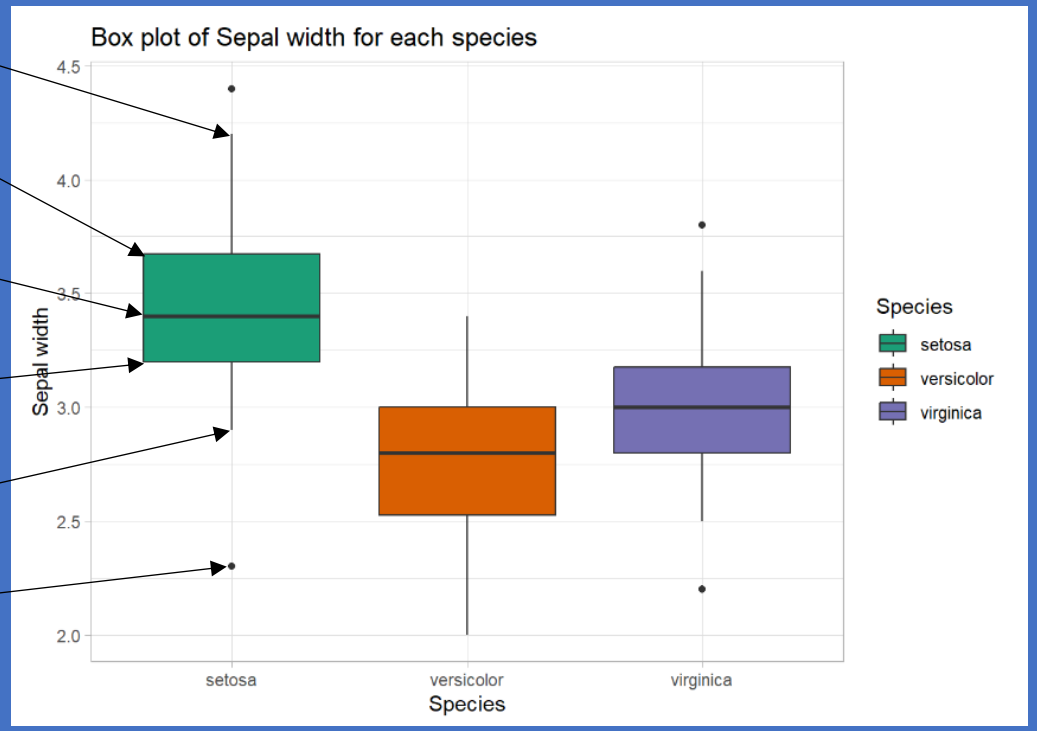
Upper quartile

Median

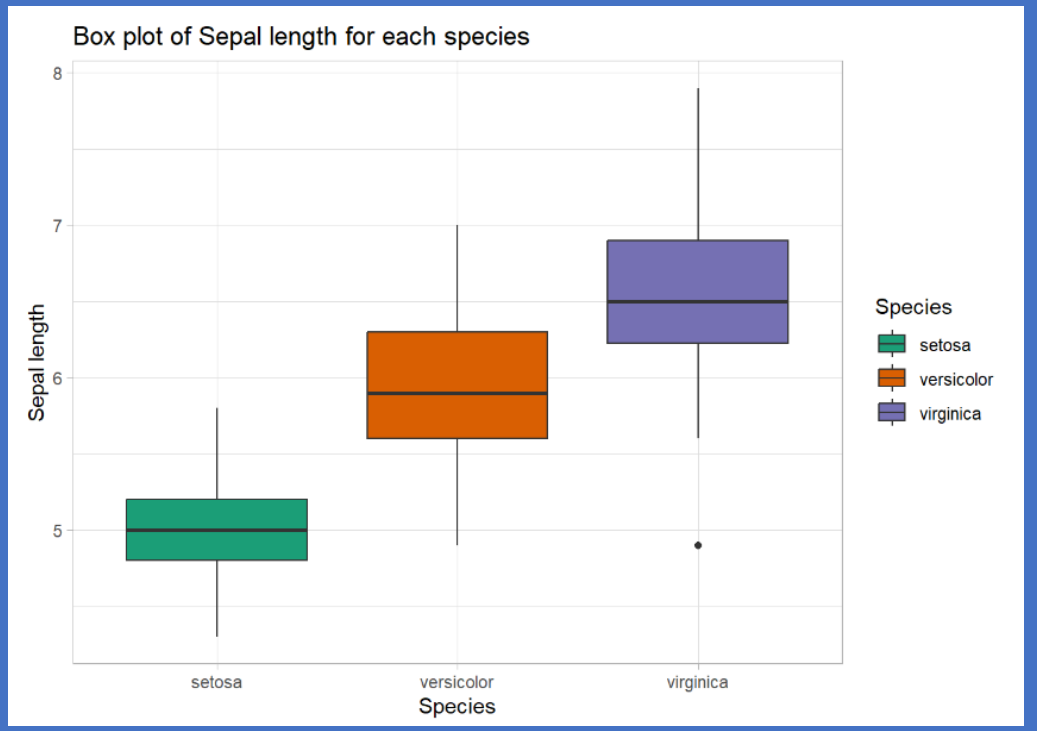
Lower quartile

Minimum value

Outliers



Sepal length



Petal width and Petal length

Let’s go through the box plot of Petal width and Petal length

Chunk 8 (# Box plot of Petal width)

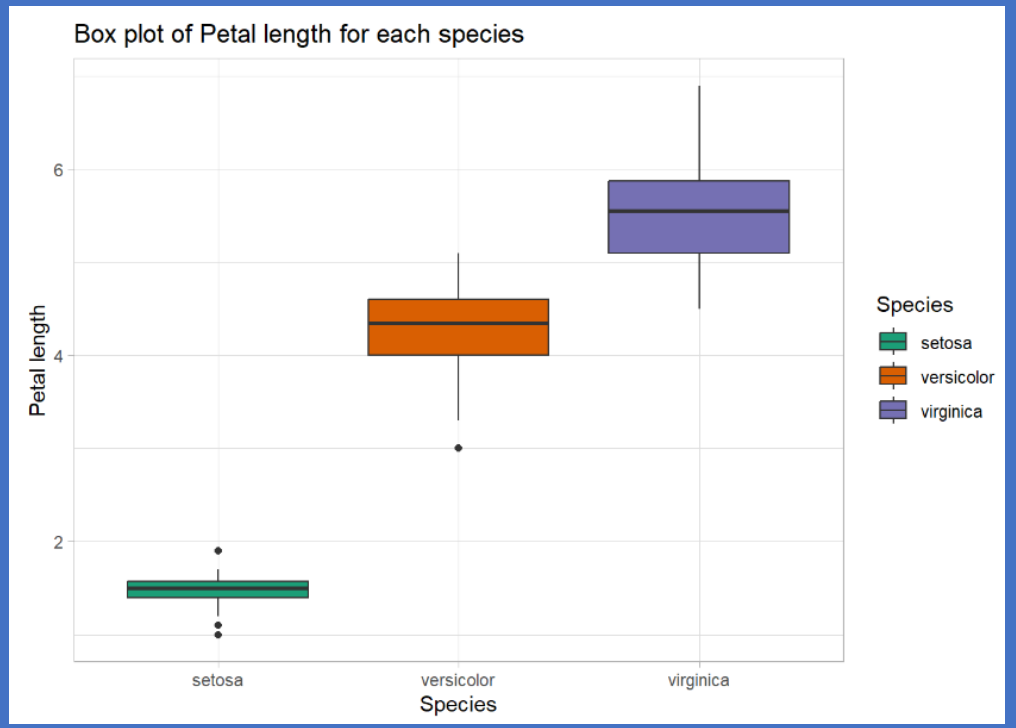
```
ggplot(data = iris,
       mapping = aes(x = Species, y = Petal.Width, fill = Species)) +
       geom_boxplot() +
       scale_fill_brewer(palette = 'Dark2') +
       theme_light() +
       labs(title = 'Box plot of Petal width for each species',
            x = 'Species', y = 'Petal width')
```

Chunk 9 (# Box plot of Petal length)

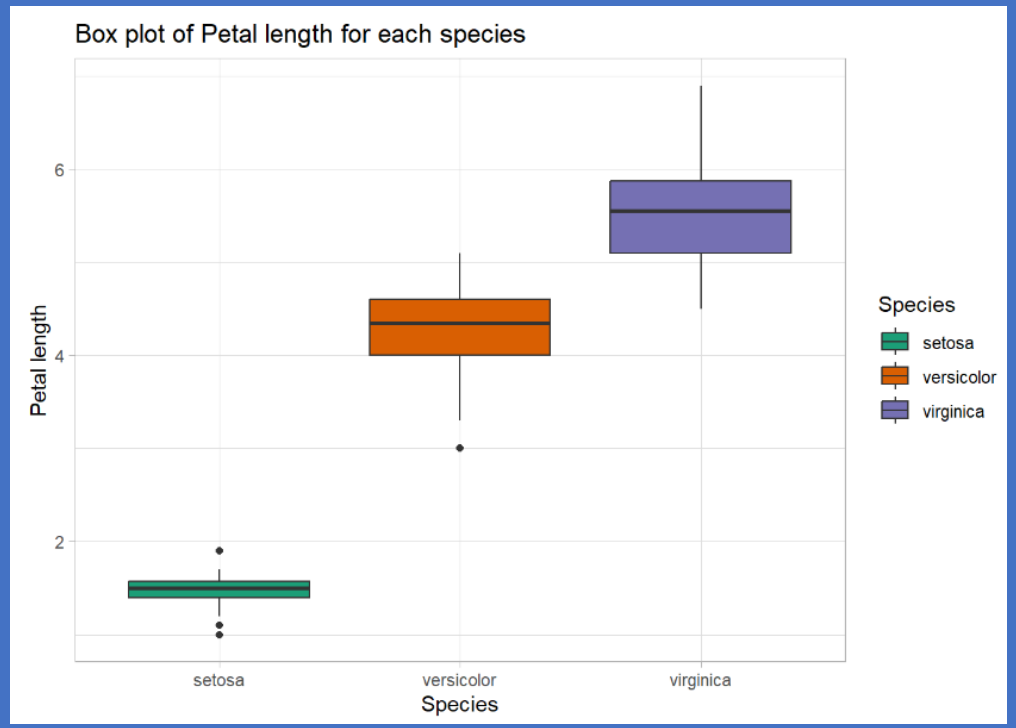
```
ggplot(data = iris,
       mapping = aes(x = Species, y = Petal.Length, fill = Species)) +
       geom_boxplot() +
       scale_fill_brewer(palette = 'Dark2') +
       theme_light() +
       labs(title = 'Box plot of Petal length for each species',
            x = 'Species', y = 'Petal length')
```

Sepal width and Sepal length

Petal width



Petal length

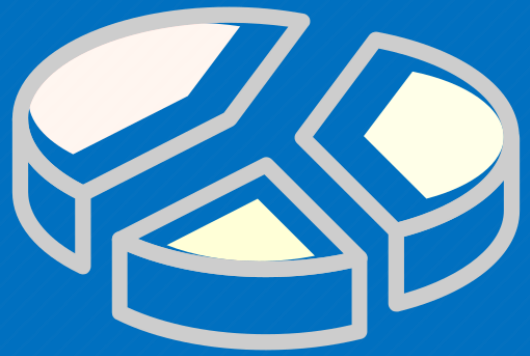


What insights can we draw from this?

- Petal-related variables (petal length and petal width) seem to be better predictors for distinguishing between the three species.
- This suggests that petal-based measurements are likely to be the most effective features for machine learning algorithms when classifying species in the Iris dataset.

- Sepal-related variables (sepal width and sepal length) are less effective due to overlapping ranges, especially between Versicolor and Virginica.
- Which means, the sepal measurements may not provide the same level of predictive power as petal measurements.

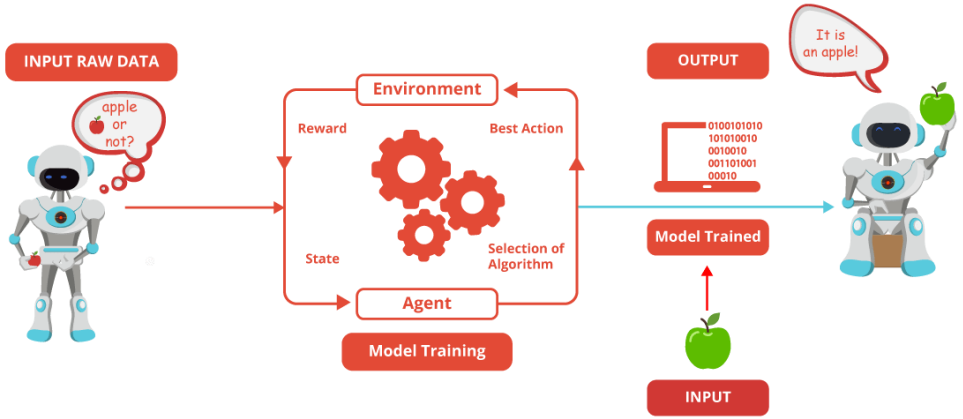
DATA PARTITIONING



Before we begin to train our machine learning model, we need to divide our data set into **train** and **test** subset.

Train data set: A training dataset is a collection of examples used to train a model to make predictions or classifications.

Test data set: A testing dataset is a set of data used to evaluate a model's performance after it has been trained.



Let’s begin our data partitioning

Chunk 10

First, lets set a random seed. This will ensure that we will be able to replicate our results when we rerun our analysis.

```
set.seed(222)
```

Chunk 11

Next, we will create train/test partition with `createDataPartition()` from the CARET package.

```
train_index <- createDataPartition(y = iris$Species, # y = dependent variable.
                                   p = .7, # Species split into 70% training set
                                           and 30% testing set
                                   list = FALSE, #  Sets results to matrix form
                                   times = 1) # Sets number of partitions to be
                                                created, which is 1 in this case
```

So here our iris data set is split into 70% training set. So naturally our testing set will be 30%

Let’s begin our data partitioning

Chunk 12

Now we can split our data into train and test data using the randomly sampled `train_index` that we just formed.

```
train_data <- iris[train_index,] # Use train_index of iris data set to create
                                   train_data
test_data <- iris[-train_index,] # Use whatever the is not in train_index to create
                                   test_data
```

Our data partitioning is successfully done!!

MACHINE LEARNING

It is finally time to create machine learning models in order to predict which category of species (setosa, versicolor, virginica) each iris flower belongs to.

Decision tree

Quick recap:

- A decision tree is a machine learning algorithm that uses a tree-like structure to classify data or predict outcomes.
- A decision tree uses input variables to create branches of decisions to ultimately derive a prediction.

Chunk 13

```
# Evaluate the Decision tree model with a 10-fold cross validation
fitControl <- trainControl(method = "cv", number = 10, savePredictions = TRUE)
```

Cross validation: The dataset is devided into 10 equal sized folds(subsets), which means the decision tree model is trained on 9 training folds and then evaluated on the 1 test fold.

Why 10 times? : Because it is more reliable estimate, and uses the data efficiently.

Decision tree

Chunk 14

Create a predictor model with `train()` from CARET package. Specify `method = 'rpart'` to run the decision tree model

```
# Create model
dt_model <- train(Species ~ ., # Set Y variable followed by '~'. The period
                                 indicates to include all variables for prediction.
                  data = train_data, # Data
                  method = 'rpart', # Specify SVM model
                  trControl = fitControl) # Use cross validation
```

Check the predicted accuracy of our decision tree model by running it on resamples of our train data.

```
confusionMatrix(dt_model)
```

A confusion matrix is a table that shows how well a machine learning model predicts the classification of a dataset. It's a visualization tool that compares the predicted values to the actual values.

Decision tree

Output

Cross-Validated (10 fold) Confusion Matrix

(entries are percentual average cell counts across resamples)

Prediction	setosa	versicolor	virginica
setosa	33.3	0	0
versicolor	0	29.5	5.7
virginica	0	3.8	27.6

Accuracy (average) : 0.9048

The results here tell us that our average accuracy is 90.48% when testing our data on re samples of our training data. We can also see what was predicted correctly/incorrectly.

Decision tree

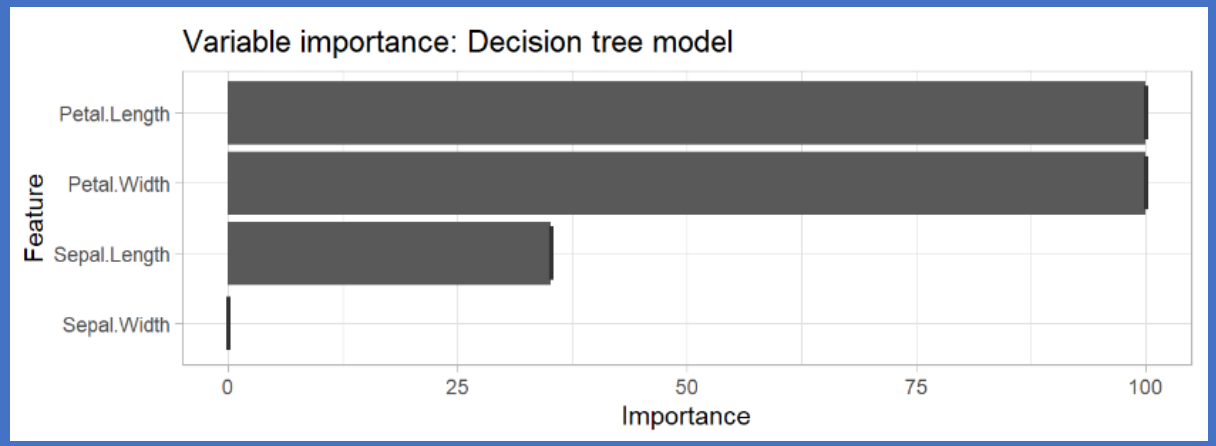
Chunk 15

Let's check the importance of each variants through a bar graph.

```
# Create object of importance of our variables
dt_importance <- varImp(dt_model)

# Create plot of importance of variables
ggplot(data = dt_importance, mapping = aes(x = dt_importance[,1])) + # Data & mapping
  geom_boxplot() + # Create box plot
  labs(title = "Variable importance: Decision tree model") + # Title
  theme_light() # Theme
```

Output



As we discussed earlier Petal length and Petal width show significant importance in classification

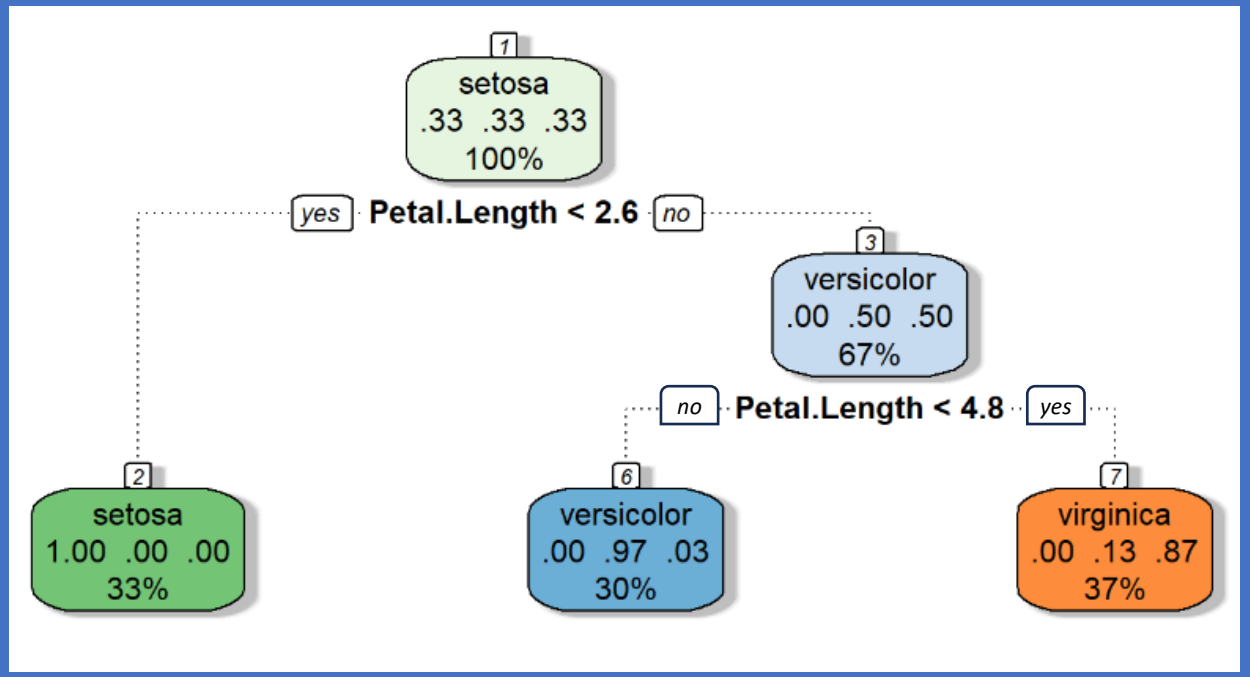
Decision tree

Chunk 16

Let's plot the decision tree using `fancyRpartPlot()` from the RATTLE package. This will give us clear insight into how the model makes its predictions.

```
fancyRpartPlot(dt_model$finalModel, # Accesses the underlying decision tree model
               sub = '') # Specifies a subtitle for the plot.
```

Output



- If petal length is less than 2.6 it is setosa.
- If petal length is in between 2.6 and 4.8 it is virginica.
- If petal length is in greater than 4.8 it is versicolor.

As we can see petal length was the only variable that was needed to predict the model.

Decision tree

Prediction using test data

Let's predict with the test data

Chunk 17 # Use the created `dt_model` to run a prediction on the test data.

```
prediction_dt <- predict(dt_model, test_data)
```

Chunk 18 # Check the proportions of the predictions which are accurate.

```
# Generate the prediction table
confusion_matrix <- table(prediction_dt, test_data$Species) %>% # Create prediction
                                                                table.
                    prop.table() %>% # Convert table values into proportions
                                     instead of counts.
                    round(2) # Round numbers to 2 significant values.

accuracy <- sum(diag(confusion_matrix)) / sum(confusion_matrix) # Calculate accuracy

confusion_matrix # Display the matrix
```

Decision tree

Output

Cross-Validated (10 fold) Confusion Matrix

(entries are percentual average cell counts across resamples)

prediction_dt	setosa	versicolor	virginica
setosa	0.33	0	0
versicolor	0	0.31	0
virginica	0	0.02	0.33

Accuracy (average) : 0.979798

As we can see, almost 98% of the species classification predicted correctly.

Logistic Regression

Quick recap:
- The goal is to predict the probability that an instance belongs to a given class or not.
- Logistic regression is a statistical algorithm which analyze the relationship between two data factors.

We will create a binary classification problem by grouping the species into “setosa” and “non-setosa”.

Create a Binary Variable

Chunk 19

#We start by creating a new column in the Iris dataset, called BinarySpecies. This column will label a flower as setosa if its species is setosa, or non-setosa otherwise.

```
# Create a binary species column
iris$BinarySpecies <- ifelse(iris$Species == "setosa", "setosa", "non-setosa")
```

Logistic Regression

Chunk 20 #Logistic regression requires the target variable to be a factor (categorical data).

```
# Create a binary species column
iris$BinarySpecies <- as.factor(iris$BinarySpecies)
```

Chunk 21 # Add BinarySpecies to train_data and test_data based on their row names

```
train_data$BinarySpecies <- iris[row.names(train_data), "BinarySpecies"]
test_data$BinarySpecies <- iris[row.names(test_data), "BinarySpecies"]
```

Logistic Regression

Chunk 22

Now, let's build the logistic regression model using `glm()` the function. The goal is to predict the BinarySpecies column using the flower's measurements (sepal length, sepal width, petal length, and petal width).

```
log_model <- glm(BinarySpecies ~ Sepal.Length + Sepal.Width + Petal.Length
+Petal.Width, # Response variable and predictor variables
                 data = train_data,
                 family = binomial) # Assumes a binary outcome
```

Here `BinarySpecies` is the response variable (dependent variable), which the outcome being predicted. It is assumed to be a binary variable (e.g., 0 or 1) because of the `family = binomial` argument.

Model summary

To see how well the model fits the training data, we use the `summary()` function.

```
summary(log_model)
```

Logistic Regression

Output

```
##
## Call:
## glm(formula = BinarySpecies ~ Sepal.Length + Sepal.Width + Petal.Length +
##     Petal.Width, family = binomial, data = train_data)
##
## Coefficients:
##                Estimate Std. Error z value Pr(>|z|)
## (Intercept)      -6.736 363100.801       0        1
## Sepal.Length      6.729 104951.965       0        1
## Sepal.Width      16.110  90634.117       0        1
## Petal.Length    -36.667 125567.319       0        1
## Petal.Width      32.964 250997.843       0        1
##
## (Dispersion parameter for binomial family taken to be 1)
##
##     Null deviance: 1.3367e+02  on 104  degrees of freedom
## Residual deviance: 1.6928e-09  on 100  degrees of freedom
## AIC: 10
##
## Number of Fisher Scoring iterations: 25
```

Logistic Regression

Visualizing the Model's Predictions

We can plot the logistic regression curve for one of the most significant predictors to understand how it distinguishes between "setosa" and "non-setosa."

Chunk23

```
ggplot(data = train_data, aes(x = Petal.Length, y = as.numeric(BinarySpecies == "setosa"), color = BinarySpecies)) + # Map Petal.Length to x-axis and binary species to y-axis, coloring by species.
```

```
geom_point(size = 3, alpha = 0.7) + # Add scatter points with size and transparency.
```

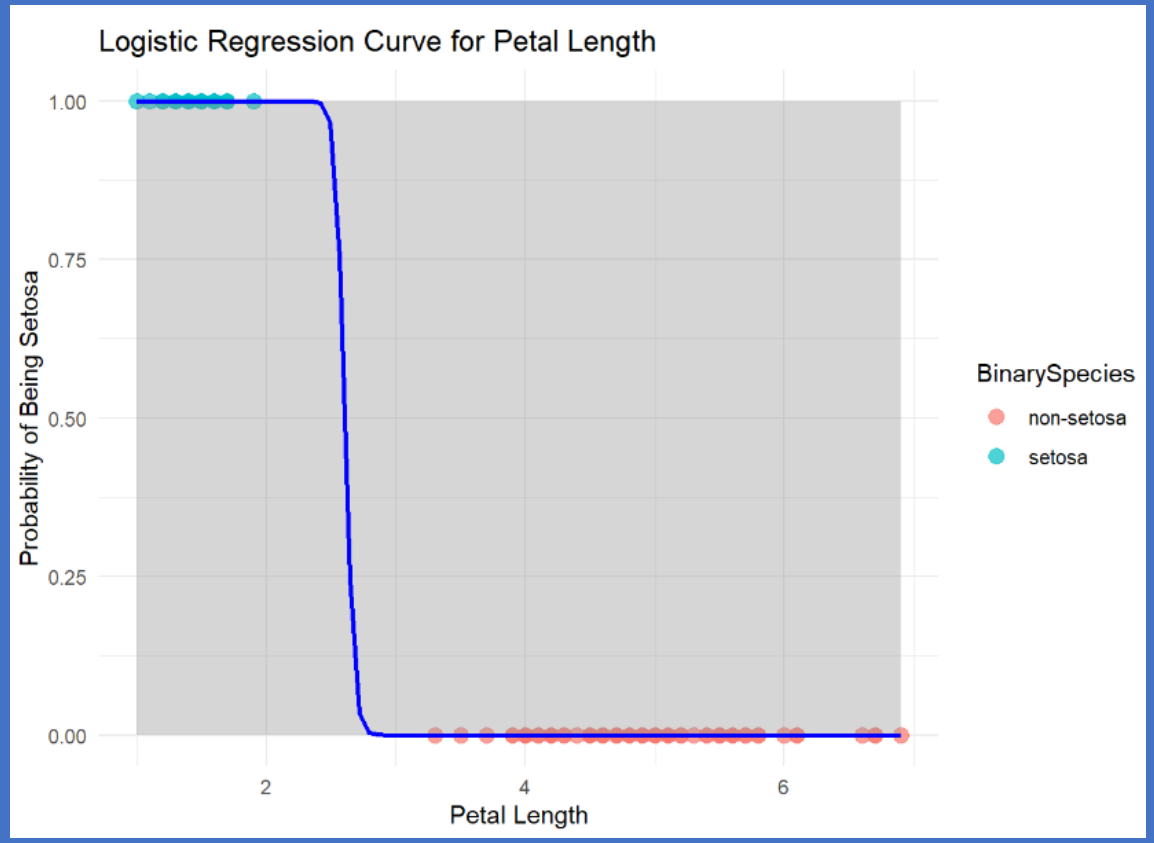
```
stat_smooth(method = "glm", method.args = list(family = "binomial"), color = "blue") + # Add a logistic regression curve in blue.
```

```
labs(title = "Logistic Regression Curve for Petal Length", x = "Petal Length", y = "Probability of Being Setosa") + # Add title and axis labels.
```

```
theme_minimal() # Apply a minimal theme to the plot.
```

Logistic Regression

Output



- Blue Curve: Represents the logistic regression model's predicted probability of being classified as "setosa" based on Petal.Length.
- Points: Show actual classifications (1 for "setosa," 0 for "non-setosa").
- Trend: A sharp increase in the probability of "setosa" classification is observed as Petal.Length decreases, which aligns with our understanding of the data.

Logistic Regression

Make Predictions on the Test Data

Chunk 24

We use the `predict()` function to generate predictions on the test dataset.

```
log_preds <- predict(log_model, test_data, type = "response")
```

The `type = "response"` option gives probabilities (values between 0 and 1) for the predicted class.

Convert Probabilities to Class Labels

We classify the predictions as setosa if the probability is greater than 0.5, and non-setosa otherwise.

Chunk 25

```
log_class <- ifelse(log_preds > 0.5, "setosa", "non-setosa") # Threshold is set at
0.5 to classify predictions as 'setosa' or 'non-setosa'

log_class <- as.factor(log_class) # Converts the predicted labels to factors for
comparison with actual labels.
```

Logistic Regression

Evaluate the model

To evaluate the model, we compare the predicted labels with the actual labels in the test dataset using a confusion matrix.

Chunk 26

```
# A confusion matrix is created to compare predicted and actual
conf_matrix_log <- table(Predicted = log_class, Actual = test_data$BinarySpecies
labels)

conf_matrix_log
```

Logistic Regression

Output

Predicted	non-setosa	setosa
non-setosa	30	0
setosa	0	15

Interpretation

From the table above table:

- Predicted as “setosa” and actually “setosa” (True Positives): 30
- Predicted as “non-setosa” and actually “non-setosa” (True Negatives): 15
- Predicted as “setosa” but actually “non-setosa” (False Positives): 0
- Predicted as “non-setosa” but actually “setosa” (False Negatives): 0

Machine learning with R

Yes! You can do ML in R

You Don't Need to Be an Expert to Get Started!

- No Need to Understand Everything Right AwayRun through the R Markdown file and focus on getting results!
- Use R's built-in help (?FunctionName) to explore what each function does.

You Don't Need to Know How Algorithms Work

- Focus on running them and interpreting the output.
- Learn the limitations and how to configure them later.

You Don't Need to Be a Programmer

- Functions and assignments in R are simple
- Follow the structure of the provided Markdown file.

You Don't Need to Be a Machine Learning Expert

- Start by loading data, running models, and evaluating results.
- Understand accuracy and cross-validation over time.

- Your First ML Project in R
- Follow the steps in the R Markdown file:
  - Load data
  - Explore and evaluate algorithms
  - Make predictions
- Focus on learning the process, not perfection!
- Additional resources and tutorials will deepen your knowledge.
- It's simple: If you can click and follow, you can do ML in R!