Iris_Machine_Learning_Project

AUTHOR
Abhinav_Maheshwaram

Machine Learning with Iris Dataset

Introduction:

The Iris Machine Learning Project focuses on exploring different algorithms to analyze the Iris dataset. This report delves into the data exploration, transformation, and analysis phases, comparing linear regression, polynomial regression, and a random forest model.

Data Exploration

Loading Necessary libraries

```
library(ggplot2) # Data visualization
library(plotly) # Interactive data visualizations
```

```
Attaching package: 'plotly'

The following object is masked from 'package:ggplot2':

    last_plot

The following object is masked from 'package:stats':

    filter

The following object is masked from 'package:graphics':

    layout
```

```
library(psych) # Correlation visualizations
```

```
Attaching package: 'psych'

The following objects are masked from 'package:ggplot2':

    %+%, alpha
```

```
library(rattle) # Graphing decision trees
```

```
Loading required package: tibble

Loading required package: bitops

Rattle: A free graphical interface for data science with R.
Version 5.5.1 Copyright (c) 2006-2021 Togaware Pty Ltd.
Type 'rattle()' to shake, rattle, and roll your data.
```

```
library(caret) # Machine learning
```

```
Loading required package: lattice

Warning in system("timedatectl", intern = TRUE): running command 'timedatectl'
had status 1
```

```
library(GGally) # Exploratory data analysis
```

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

```
library(caTools) # Data splitting
library(dplyr) # Data manipulation
```

```
Attaching package: 'dplyr'

The following objects are masked from 'package:stats':

    filter, lag

The following objects are masked from 'package:base':

    intersect, setdiff, setequal, union
```

```
library(caret) # Streamlining machine learning processes
library(e1071) # Additional machine learning support
```

Iris is an inbuilt data set with R. We will be using Iris data set to analyze linear regression model

First step is to load the Iris Dataset

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

To have a look at the data, There are several options to do so.

We can simply use head command to see the number of fields

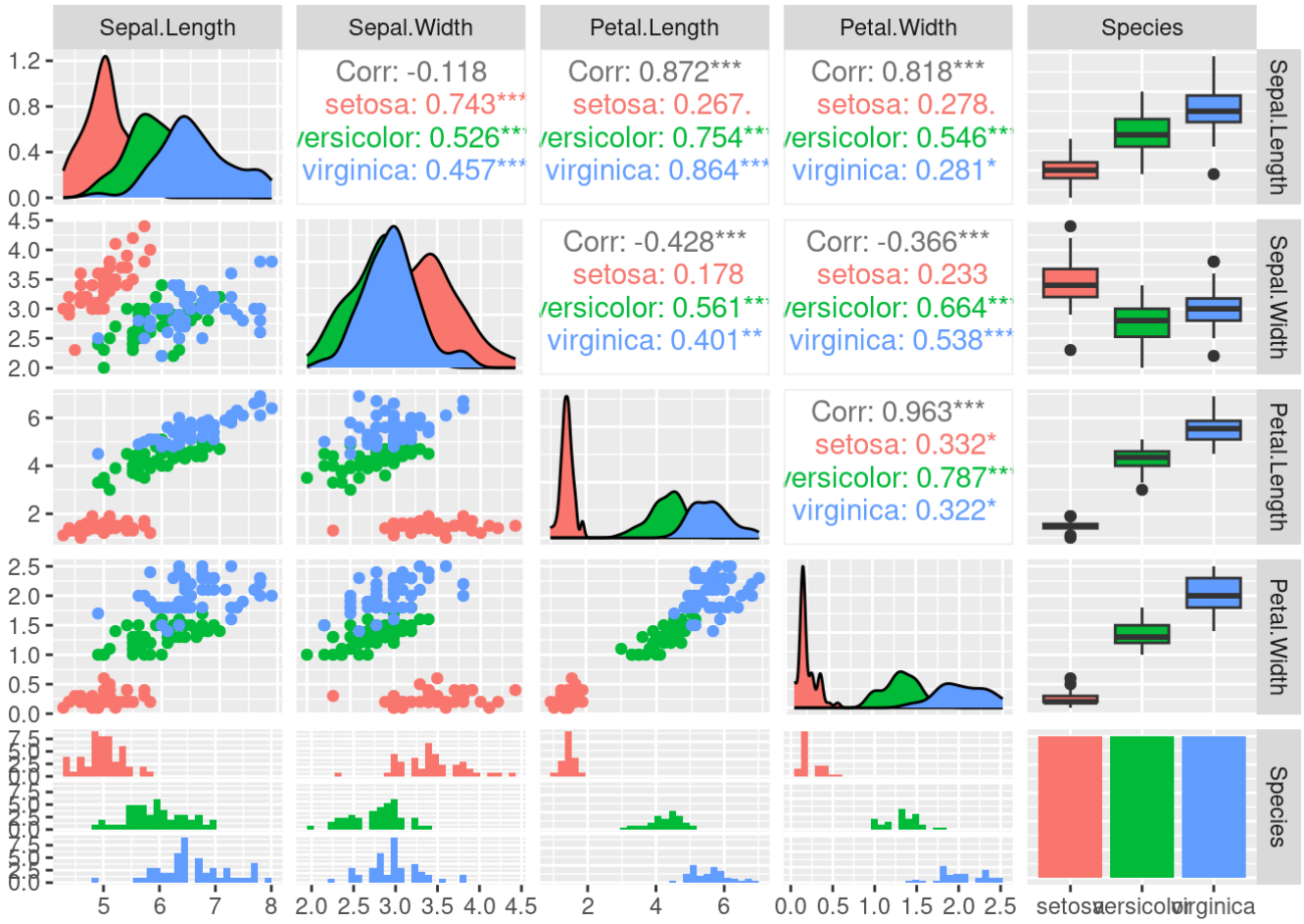
```
head(iris)
```

```
  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
```

Correlation Analysis:

```
ggpairs(iris, 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`.
```



The correlation between the variables is displayed in the upper section of the correlation matrix. All variables, with the exception of sepal length and width, have moderate to strong correlations. The bottom half of the matrix, which uses colors to split the data points by iris species, also provides us with further information about the scatter plots of these correlations. This enables us to observe the species-specific clusters that exist.

Data Transformation

`set.seed()` function is used to set the seed for the random number generator. The random number generator in R is based on an initial seed value, and setting the seed ensures reproducibility of random processes.

```
set.seed(200)
```

Splitting data into training and test sets is crucial for evaluating a machine learning model's performance on new, unseen data, preventing overfitting, assessing generalization, tuning hyperparameters, and avoiding data leakage.

```
split = sample.split(iris$Species, SplitRatio = 2/3)
training_set = subset(iris, split == TRUE)
test_set = subset(iris, split == FALSE)
```

```
model <- lm(Species ~ Petal.Length + Petal.Width, data = training_set)
```

```
Warning in model.response(mf, "numeric"): using type = "numeric" with a factor
response will be ignored

Warning in Ops.factor(y, z$residuals): '-' not meaningful for factors
```

```
summary(iris)
```

```
  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
```

Data Analysis:

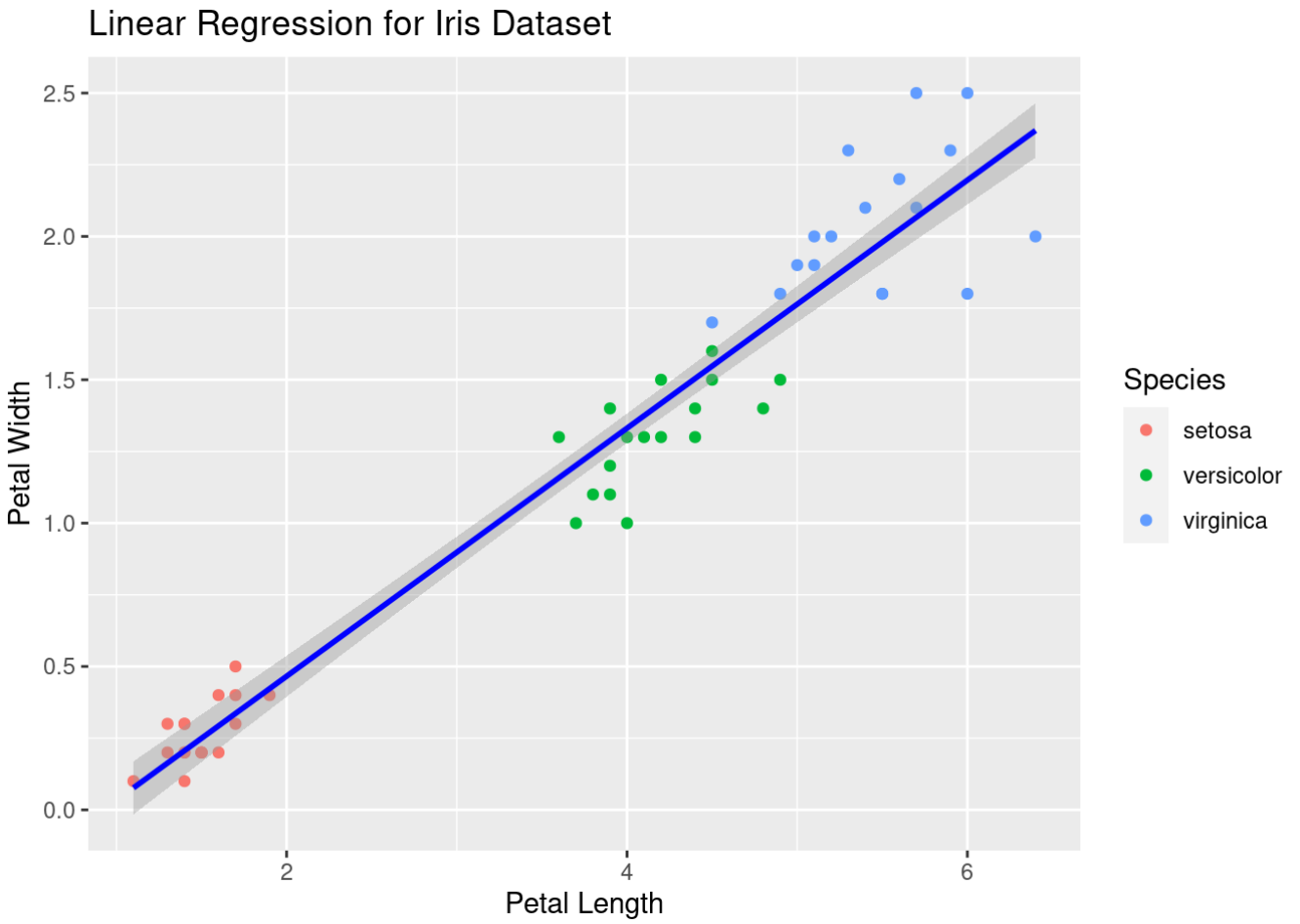
```
predictions <- predict(model, test_set)
actual <- test_set$Species
mse <- mean((predictions - actual)^2)
```

```
Warning in Ops.factor(predictions, actual): '-' not meaningful for factors
```

```
rmse <- sqrt(mse)
```

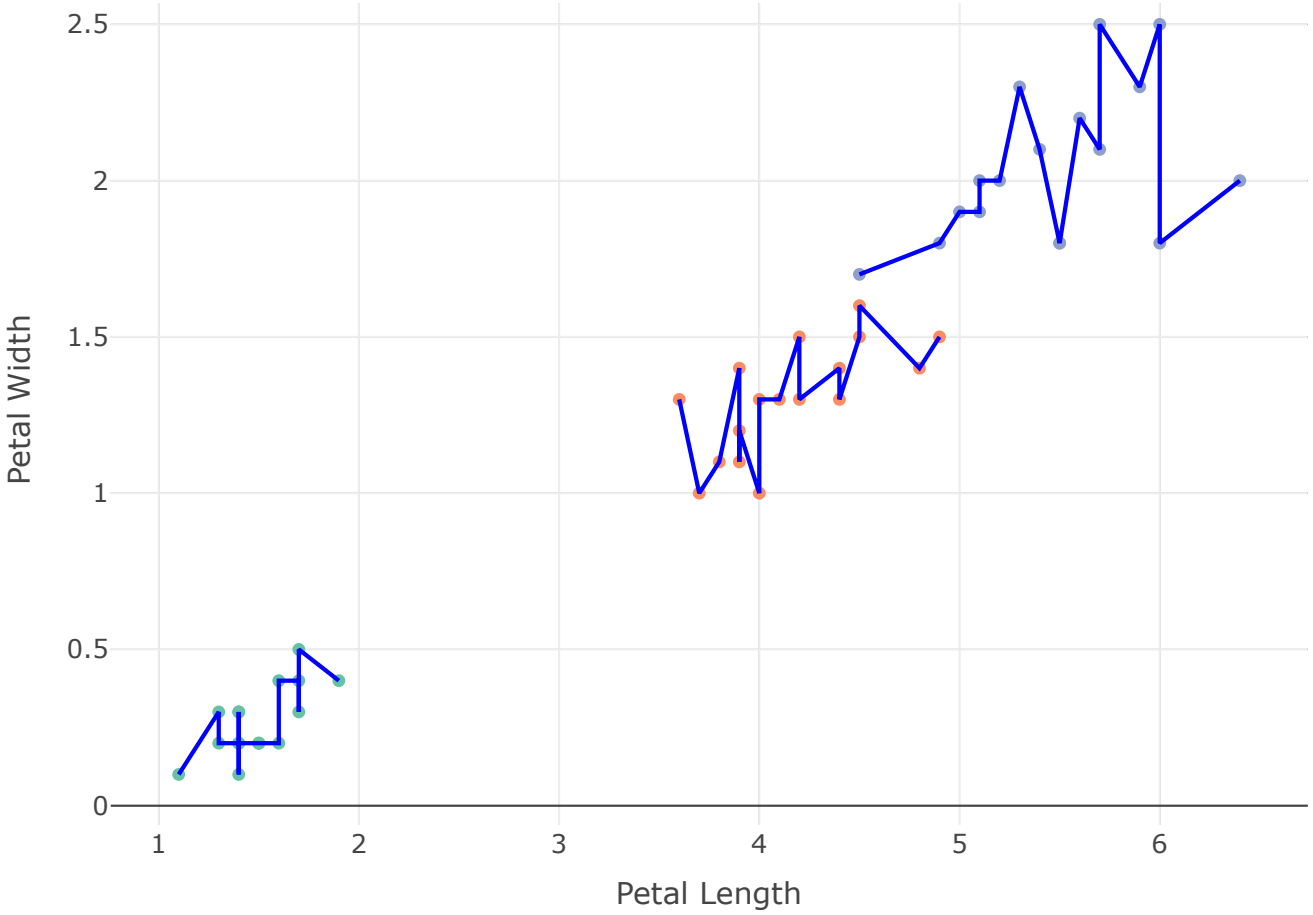
```
ggplot(test_set, aes(x = Petal.Length, y = Petal.Width, color = Species)) +
  geom_point() +
```

```
  geom_smooth(method = "lm", formula = y ~ x, color = "blue") +
  labs(title = "Linear Regression for Iris Dataset",
       x = "Petal Length",
       y = "Petal Width")
```




```
# Create a scatter plot with linear regression line
plot <- plot_ly(test_set, x = ~Petal.Length, y = ~Petal.Width, color = ~Species)
  add_markers() %>%
  add_lines(type = "lm", line = list(color = "blue")) %>%
  layout(title = "Linear Regression for Iris Dataset",
         xaxis = list(title = "Petal Length"),
         yaxis = list(title = "Petal Width"),
         showlegend = TRUE)

# Display the plot
plot
```



Machine Learning Using Linear Regression Method:

```
# Create and train the logistic regression model
model <- glm(Species ~ Petal.Length + Petal.Width, data = training_set, family =
```

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

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

```
# Predict the results on the testing set
predicted <- predict(model, newdata = test_set, type = "response")

# Convert predicted probabilities to predicted class labels
predicted_class <- ifelse(predicted > 0.5, "versicolor", "setosa")

# View the predicted class labels
predicted_class
```

```
        4         5         6         7        12        14
"setosa" "setosa" "setosa" "setosa" "setosa" "setosa"
```

```
         19          24          27          35          38          42
  "setosa"    "setosa"    "setosa"    "setosa"    "setosa"    "setosa"
         43          45          46          49          50          53
  "setosa"    "setosa"    "setosa"    "setosa"    "setosa" "versicolor"
         60          62          63          65          66          70
"versicolor" "versicolor" "versicolor" "versicolor" "versicolor" "versicolor"
         77          79          81          82          83          86
"versicolor" "versicolor" "versicolor" "versicolor" "versicolor" "versicolor"
         88          90          95         100         101         102
"versicolor" "versicolor" "versicolor" "versicolor" "versicolor" "versicolor"
        107         111         116         117         124         125
"versicolor" "versicolor" "versicolor" "versicolor" "versicolor" "versicolor"
        126         132         133         138         140         144
"versicolor" "versicolor" "versicolor" "versicolor" "versicolor" "versicolor"
        145         147         148
"versicolor" "versicolor" "versicolor"
```

```
# Calculate accuracy and confusion matrix
actual_class <- test_set$Species
accuracy <- mean(predicted_class == actual_class) * 100
cat("Accuracy of the model is", round(accuracy, 2), "%\n")
```

```
Accuracy of the model is 66.67 %
```

```
# Create the confusion matrix
confusionMatrix(data = factor(predicted_class), reference = factor(actual_class)
```

```
Warning in levels(reference) != levels(data): longer object length is not a
multiple of shorter object length

Warning in confusionMatrix.default(data = factor(predicted_class), reference =
factor(actual_class)): Levels are not in the same order for reference and data.
Refactoring data to match.

Confusion Matrix and Statistics

            Reference
Prediction   setosa versicolor virginica
  setosa         17          0         0
  versicolor      0         17        17
  virginica       0          0         0

Overall Statistics
```

```
               Accuracy : 0.6667
                 95% CI : (0.5208, 0.7924)
    No Information Rate : 0.3333
    P-Value [Acc > NIR] : 1.178e-06

                  Kappa : 0.5

 Mcnemar's Test P-Value : NA

Statistics by Class:

                     Class: setosa Class: versicolor Class: virginica
Sensitivity                 1.0000            1.0000           0.0000
Specificity                 1.0000            0.5000           1.0000
Pos Pred Value              1.0000            0.5000              NaN
Neg Pred Value              1.0000            1.0000           0.6667
Prevalence                  0.3333            0.3333           0.3333
Detection Rate              0.3333            0.3333           0.0000
Detection Prevalence        0.3333            0.6667           0.0000
Balanced Accuracy           1.0000            0.7500           0.5000
```

Machine Learning Using Polynomial Regression Method:

```
# Create and train the logistic regression model with polynomial terms
model <- glm(Species ~ Petal.Length + Petal.Width + Petal.Length + Petal.Width,
```

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

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

```
# Predict the results on the testing set
predicted <- predict(model, newdata = test_set, type = "response")

# Convert predicted probabilities to predicted class labels
predicted_class <- ifelse(predicted > 0.5, "versicolor", "setosa")

# View the predicted class labels
predicted_class
```

```
        4         5         6         7        12        14 
"setosa" "setosa" "setosa" "setosa" "setosa" "setosa" 
       19        24        27        35        38        42 
"setosa" "setosa" "setosa" "setosa" "setosa" "setosa" 
```

```
          43           45           46           49           50           53
    "setosa"     "setosa"     "setosa"     "setosa"     "setosa" "versicolor"
          60           62           63           65           66           70
"versicolor" "versicolor" "versicolor" "versicolor" "versicolor" "versicolor"
          77           79           81           82           83           86
"versicolor" "versicolor" "versicolor" "versicolor" "versicolor" "versicolor"
          88           90           95          100          101          102
"versicolor" "versicolor" "versicolor" "versicolor" "versicolor" "versicolor"
         107          111          116          117          124          125
"versicolor" "versicolor" "versicolor" "versicolor" "versicolor" "versicolor"
         126          132          133          138          140          144
"versicolor" "versicolor" "versicolor" "versicolor" "versicolor" "versicolor"
         145          147          148
"versicolor" "versicolor" "versicolor"
```

```
# Calculate accuracy and confusion matrix
actual_class <- test_set$Species
accuracy <- mean(predicted_class == actual_class) * 100
cat("Accuracy of the polynomial logistic regression model is", round(accuracy, 2
```

```
Accuracy of the polynomial logistic regression model is 66.67 %
```

```
# Create the confusion matrix
confusionMatrix(data = factor(predicted_class), reference = factor(actual_class)
```

```
Warning in levels(reference) != levels(data): longer object length is not a
multiple of shorter object length

Warning in confusionMatrix.default(data = factor(predicted_class), reference =
factor(actual_class)): Levels are not in the same order for reference and data.
Refactoring data to match.

Confusion Matrix and Statistics

            Reference
Prediction   setosa versicolor virginica
  setosa         17          0         0
  versicolor      0         17        17
  virginica       0          0         0

Overall Statistics

               Accuracy : 0.6667
                 95% CI : (0.5208, 0.7924)
```

```
    No Information Rate : 0.3333
    P-Value [Acc > NIR] : 1.178e-06

                  Kappa : 0.5

 Mcnemar's Test P-Value : NA

Statistics by Class:

                     Class: setosa Class: versicolor Class: virginica
Sensitivity                 1.0000            1.0000           0.0000
Specificity                 1.0000            0.5000           1.0000
Pos Pred Value              1.0000            0.5000              NaN
Neg Pred Value              1.0000            1.0000           0.6667
Prevalence                  0.3333            0.3333           0.3333
Detection Rate              0.3333            0.3333           0.0000
Detection Prevalence        0.3333            0.6667           0.0000
Balanced Accuracy           1.0000            0.7500           0.5000
```

Machine Learning Using Random Forest Method:

```
library(randomForest)
```

```
randomForest 4.7-1.1

Type rfNews() to see new features/changes/bug fixes.


Attaching package: 'randomForest'

The following object is masked from 'package:dplyr':

    combine

The following object is masked from 'package:rattle':

    importance

The following object is masked from 'package:psych':

    outlier

The following object is masked from 'package:ggplot2':
```

```
    margin
```

```
# Create and train the random forest model
model_rf <- randomForest(Species ~ Petal.Length + Petal.Width, data = training_s

# Predict the results on the testing set
predicted_rf <- predict(model_rf, newdata = test_set)

# Calculate accuracy
accuracy_rf <- mean(predicted_rf == actual_class) * 100
cat("Accuracy of the random forest model is", round(accuracy_rf, 2), "%\n")
```

```
Accuracy of the random forest model is 98.04 %
```

Conclusion:

Linear and Polynomial Regression:

1. Algorithm Suitability: Linear and polynomial regression may not be suitable for the Iris dataset, which involves multi-class classification. They are typically used for regression tasks or binary classification.

2. Feature Representation: The features chosen (Petal.Length and Petal.Width) might not capture the underlying patterns in the dataset effectively, leading to lower accuracy.

Random Forest:

1. Algorithm Choice: Random Forest, a more complex ensemble learning algorithm, outperformed linear and polynomial regression. It is better suited for multi-class classification tasks.

2. Ensemble Learning: Random Forest combines multiple decision trees to improve accuracy and generalization. This ensemble approach proved effective in capturing complex relationships in the Iris dataset.

3. Hyperparameter Tuning: The default parameters of the Random Forest model might have been well-suited for the Iris dataset, or hyperparameter tuning could have played a role in achieving the high accuracy.

Key Takeaways:

- The choice of algorithm is crucial, and different algorithms perform differently on various

datasets.

- Ensemble methods like Random Forest can handle complex relationships and provide better accuracy for certain tasks.

- Feature selection and engineering play a significant role in model performance.

- Experimentation and trying multiple algorithms are essential in finding the most suitable approach for a specific classification problem.