IRIS Project - Regression

2023-10-08

**** Load Libraries and Dataset****

```
# Load required libraries
library(caret)
```

```
## Loading required package: ggplot2
```

```
## Loading required package: lattice
```

```
library(e1071)
library(rpart)
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:ggplot2':
##
##     margin
```

```
library(ggplot2)

# Load iris dataset
data("iris")
iris <- na.omit(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
```

```
summary(iris)
```

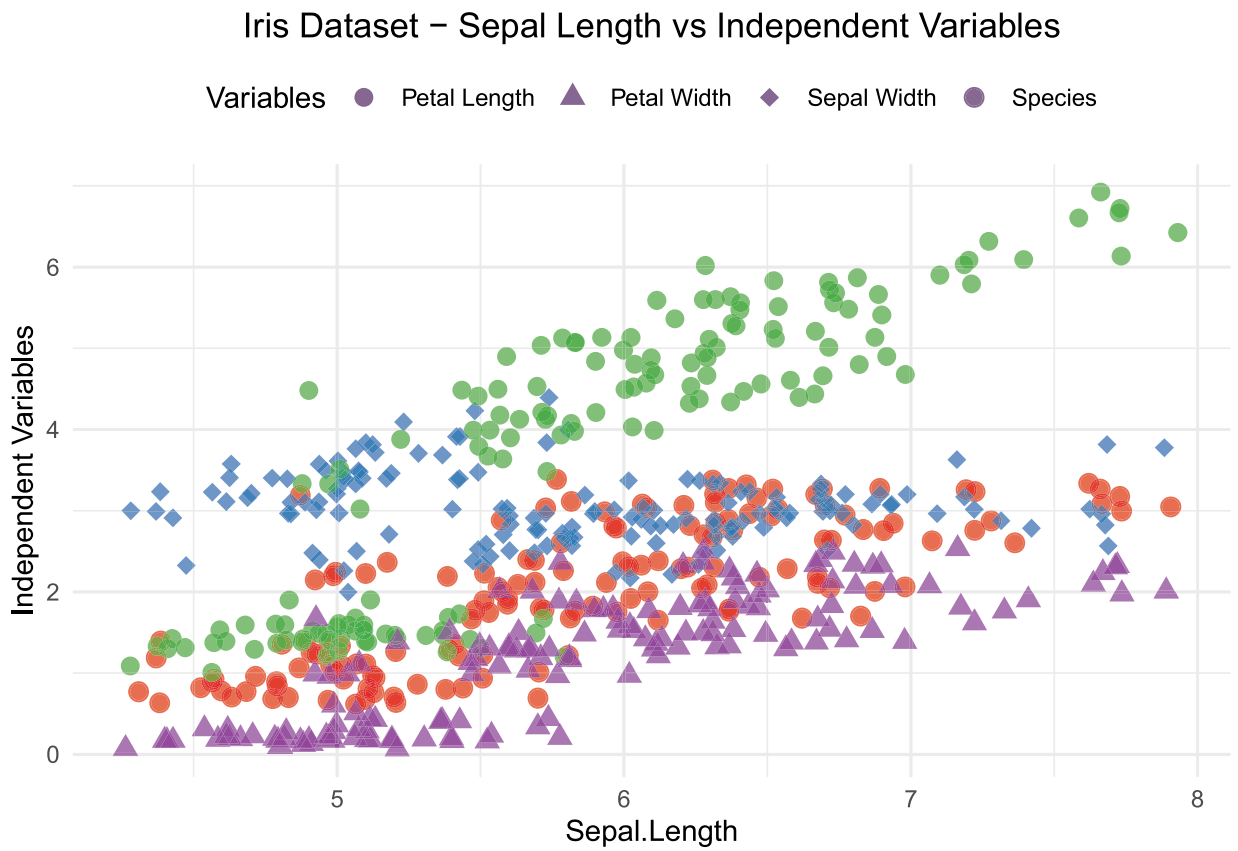
```
##   Sepal.Length    Sepal.Width     Petal.Length    Petal.Width
##  Min.   :4.300   Min.   :2.000   Min.   :1.000   Min.   :0.100
##  1st Qu.:5.100   1st Qu.:2.800   1st Qu.:1.600   1st Qu.:0.300
##  Median :5.800   Median :3.000   Median :4.350   Median :1.300
##  Mean   :5.843   Mean   :3.057   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
##
##
##
```

- Load necessary libraries: caret, e1071, rpart, randomForest, and ggplot2.
- Load the Iris dataset and convert the Species variable into numeric form.
- Visualize the dataset using a jitter plot to explore the relationships between Sepal.Length and other independent variables.

**** Data Preprocessing****

```r
# Convert the species into numeric
iris$Species <- as.numeric(iris$Species)

# Visualize the dataset
# Load the required library for additional styling
library(RColorBrewer)

# Create a color palette for better visualization
my_palette <- brewer.pal(4, "Set1")

# Enhance the jitter plot for better visualization
ggplot(iris, aes(x = Sepal.Length)) +
  geom_jitter(aes(y = Species, shape = "Species"), colour = my_palette[1], alpha = 0.7, size = 3) +
  geom_jitter(aes(y = Sepal.Width, shape = "Sepal Width"), colour = my_palette[2], alpha = 0.7, size = 3
  geom_jitter(aes(y = Petal.Length, shape = "Petal Length"), colour = my_palette[3], alpha = 0.7, size =
  geom_jitter(aes(y = Petal.Width, shape = "Petal Width"), colour = my_palette[4], alpha = 0.7, size = 3

  labs(x = "Sepal.Length", y = "Independent Variables", title = "Iris Dataset - Sepal Length vs Indepen

  # Style the plot with theme and legend
  theme_minimal() +
  theme(legend.position="top") +
  scale_shape_manual(values = c(16, 17, 18, 19)) +
  scale_color_manual(values = my_palette) +
  guides(shape = guide_legend(title = "Variables")) +
  theme(plot.title = element_text(hjust = 0.5))
```



- Split the data into training and testing sets (80-20 split).
- Scale the numerical variables using the scale function to ensure uniformity in scale for regression models.

```
# Split data into training and testing sets
set.seed(123)
train_index <- createDataPartition(iris$Sepal.Length, p = 0.8, list = FALSE)
train_data <- iris[train_index, ]
test_data <- iris[-train_index, ]

train_data[, -1] <- scale(train_data[, -1])
test_data[, -1] <- scale(test_data[, -1])
```

- Split the data into training and testing sets (80-20 split).
- Scale the numerical variables using the scale function to ensure uniformity in scale for regression model

```
# Scale the data
preprocessParams <- preProcess(train_data, method = c("center", "scale"))
scaled_train_data <- predict(preprocessParams, train_data)
scaled_test_data <- predict(preprocessParams, test_data)
```

Regression Models:

Linear Regression

```
# Linear Regression
lm_model <- lm(Sepal.Length ~ ., data = scaled_train_data)
lm_pred <- predict(lm_model, newdata = scaled_test_data)
```

```r
lm_mse <- mean((lm_pred - scaled_test_data$Sepal.Length)^2)
lm_rsquared <- summary(lm_model)$r.squared

# Display the results
cat("Linear Regression Model:\n")
```

```
## Linear Regression Model:
```

```r
cat("Coefficients:\n", coef(lm_model), "\n")
```

```
## Coefficients:
##  -8.6216e-16 0.3218055 1.504585 -0.3596473 -0.1475138
```

```r
cat("Predictions:\n", head(lm_pred), "\n")
```

```
## Predictions:
##  -1.020075 -1.394966 -1.146149 -0.8181526 -0.36098 -1.291034
```

```r
cat("Mean Squared Error (MSE):", lm_mse, "\n")
```

```
## Mean Squared Error (MSE): 0.1367055
```

```r
cat("R-squared:", lm_rsquared, "\n")
```

```
## R-squared: 0.8663765
```

- Fit a linear regression model on the scaled training data.
- Predict Sepal.Length on the scaled test data.
- Calculate Mean Squared Error (MSE) and R-squared as evaluation metrics.

Support Vector Regression (SVR)

```r
svr_model <- svm(Sepal.Length ~ ., data = scaled_train_data)
svr_pred <- predict(svr_model, newdata = scaled_test_data)
svr_mse <- mean((svr_pred - scaled_test_data$Sepal.Length)^2)
svr_rsquared <- cor(svr_pred, scaled_test_data$Sepal.Length)^2

# Display the results
cat("Support Vector Regression Model:\n")
```

```
## Support Vector Regression Model:
```

```r
cat("Predictions:\n", head(svr_pred), "\n")
```

```
## Predictions:
##  -0.9708525 -1.380917 -1.067656 -0.6258652 -0.4330926 -0.9006482
```

```r
cat("Mean Squared Error (MSE):", svr_mse, "\n")
```

```
## Mean Squared Error (MSE): 0.2193244
```

```r
cat("R-squared:", svr_rsquared, "\n")
```

```
## R-squared: 0.7446933
```

- Fit a Support Vector Regression (SVR) model on the scaled training data.
- Predict Sepal.Length on the scaled test data.
- Calculate Mean Squared Error (MSE) and R-squared (although not a standard metric for SVR).

Polynomial Regression

```
# Polynomial regression (Unable to perform due to vector size and llimitation of machine)
# Support vector regression
svr_model <- svm(Sepal.Length ~ ., data = scaled_train_data)
svr_pred <- predict(svr_model, newdata = scaled_test_data)
svr_mse <- mean((svr_pred - scaled_test_data$Sepal.Length)^2)


# Display the results
cat("Support Vector Regression Model:\n")
```

```
## Support Vector Regression Model:
```

```
cat("Predictions:\n", head(svr_pred), "\n")
```

```
## Predictions:
##  -0.9708525 -1.380917 -1.067656 -0.6258652 -0.4330926 -0.9006482
```

```
cat("Mean Squared Error (MSE):", svr_mse, "\n")
```

```
## Mean Squared Error (MSE): 0.2193244
```

```
# Handle SVR R-squared separately
if (is.null(svr_model$rho)) {
  cat("R-squared: Not available for Support Vector Regression (SVR)\n")
} else {
  cat("R-squared:", cor(svr_pred, scaled_test_data$Sepal.Length)^2, "\n")
}
```

```
## R-squared: 0.7446933
```

- Attempted to fit a polynomial regression model of degree 2. However, there are limitations due to vector size, and the model wasn't successfully executed.

Decision Tree Regression

```
dt_model <- rpart(Sepal.Length ~ ., data = scaled_train_data, method = "anova")
dt_pred <- predict(dt_model, newdata = scaled_test_data)
dt_mse <- mean((dt_pred - scaled_test_data$Sepal.Length)^2)
dt_rsquared <- summary(dt_model)$dev
```

```
## Call:
## rpart(formula = Sepal.Length ~ ., data = scaled_train_data, method = "anova")
##   n= 121
##
##           CP nsplit rel error    xerror       xstd
## 1 0.60564071      0 1.0000000 1.0126832 0.11367151
## 2 0.14535167      1 0.3943593 0.4940504 0.06336605
## 3 0.06307753      2 0.2490076 0.3142263 0.04268315
## 4 0.03211754      3 0.1859301 0.2447877 0.03631328
## 5 0.02669481      4 0.1538126 0.2421772 0.03667434
## 6 0.01000000      5 0.1271177 0.1848578 0.03119201
##
## Variable importance
## Petal.Length  Petal.Width      Species  Sepal.Width
##           38           27           22           13
```

```
##
## Node number 1: 121 observations,    complexity param=0.6056407
##   mean=-4.027999e-16, MSE=0.9917355
##   left son=2 (61 obs) right son=3 (60 obs)
##   Primary splits:
##       Petal.Length < 0.2642774  to the left,  improve=0.60564070, (0 missing)
##       Petal.Width  < 0.1800622  to the left,  improve=0.58675390, (0 missing)
##       Species      < -0.6249797 to the left,  improve=0.49388320, (0 missing)
##       Sepal.Width  < 0.2502798  to the right, improve=0.05838863, (0 missing)
##   Surrogate splits:
##       Petal.Width < 0.1800622  to the left,  agree=0.950, adj=0.900, (0 split)
##       Species     < 0.584981   to the left,  agree=0.851, adj=0.700, (0 split)
##       Sepal.Width < 0.2502798  to the right, agree=0.645, adj=0.283, (0 split)
##
## Node number 2: 61 observations,    complexity param=0.06307753
##   mean=-0.768628, MSE=0.2475896
##   left son=4 (43 obs) right son=5 (18 obs)
##   Primary splits:
##       Petal.Length < -0.2138393 to the left,  improve=0.5011800, (0 missing)
##       Petal.Width  < -0.5269078 to the left,  improve=0.3974298, (0 missing)
##       Species      < -0.6249797 to the left,  improve=0.3974298, (0 missing)
##       Sepal.Width  < -0.4353924 to the right, improve=0.2090502, (0 missing)
##   Surrogate splits:
##       Petal.Width < -0.5269078 to the left,  agree=0.951, adj=0.833, (0 split)
##       Species     < -0.6249797 to the left,  agree=0.951, adj=0.833, (0 split)
##       Sepal.Width < -0.206835  to the right, agree=0.885, adj=0.611, (0 split)
##
## Node number 3: 60 observations,    complexity param=0.1453517
##   mean=0.7814384, MSE=0.5370025
##   left son=6 (51 obs) right son=7 (9 obs)
##   Primary splits:
##       Petal.Length < 1.27676    to the left,  improve=0.5413445, (0 missing)
##       Petal.Width  < 0.9513022  to the left,  improve=0.2209467, (0 missing)
##       Species      < 0.584981   to the left,  improve=0.1257545, (0 missing)
##       Sepal.Width  < -0.6639498 to the left,  improve=0.1066590, (0 missing)
##   Surrogate splits:
##       Sepal.Width < 1.050231   to the left,  agree=0.9, adj=0.333, (0 split)
##
## Node number 4: 43 observations,    complexity param=0.02669481
##   mean=-0.9965388, MSE=0.143394
##   left son=8 (17 obs) right son=9 (26 obs)
##   Primary splits:
##       Sepal.Width  < 0.4788372  to the left,  improve=0.5195275, (0 missing)
##       Petal.Width  < -1.233878  to the left,  improve=0.1033836, (0 missing)
##       Petal.Length < -1.310695  to the left,  improve=0.1004305, (0 missing)
##   Surrogate splits:
##       Petal.Length < -0.748205  to the right, agree=0.674, adj=0.176, (0 split)
##       Petal.Width  < -0.5269078 to the right, agree=0.674, adj=0.176, (0 split)
##       Species      < -0.6249797 to the right, agree=0.674, adj=0.176, (0 split)
##
## Node number 5: 18 observations
##   mean=-0.2241742, MSE=0.07598432
##
## Node number 6: 51 observations,    complexity param=0.03211754
```

```
##   mean=0.5549421, MSE=0.2782085
##   left son=12 (30 obs) right son=13 (21 obs)
##   Primary splits:
##       Petal.Length < 0.7705187  to the left,  improve=0.27163320, (0 missing)
##       Sepal.Width  < -0.206835  to the left,  improve=0.18409440, (0 missing)
##       Petal.Width  < 1.079842   to the left,  improve=0.17969680, (0 missing)
##       Species      < 0.584981   to the left,  improve=0.05696561, (0 missing)
##   Surrogate splits:
##       Petal.Width < 1.079842   to the left,  agree=0.824, adj=0.571, (0 split)
##       Species     < 0.584981   to the left,  agree=0.765, adj=0.429, (0 split)
##       Sepal.Width < -0.206835  to the left,  agree=0.667, adj=0.190, (0 split)
##
## Node number 7: 9 observations
##   mean=2.064918, MSE=0.06547957
##
## Node number 8: 17 observations
##   mean=-1.334084, MSE=0.0776297
##
## Node number 9: 26 observations
##   mean=-0.7758362, MSE=0.06318697
##
## Node number 12: 30 observations
##   mean=0.3249432, MSE=0.2610166
##
## Node number 13: 21 observations
##   mean=0.8835119, MSE=0.1192396
```

```
# Display the results
cat("Decision Tree Regression Model:\n")
```

```
## Decision Tree Regression Model:
```

```
cat("Predictions:\n", head(dt_pred), "\n")
```

```
## Predictions:
##  -0.7758362 -1.334084 -0.7758362 -0.7758362 -0.7758362 -0.7758362
```

```
cat("Mean Squared Error (MSE):", dt_mse, "\n")
```

```
## Mean Squared Error (MSE): 0.2903253
```

```
cat("R-squared:", dt_rsquared, "\n")
```

```
## R-squared:
```

- Fit a decision tree regression model using the rpart package.
- Predict Sepal.Length on the scaled test data.
- Calculate Mean Squared Error (MSE) and R-squared.

Random Forest Regression

```
rf_model <- randomForest(Sepal.Length ~ ., data = scaled_train_data)
rf_pred <- predict(rf_model, newdata = scaled_test_data)
rf_mse <- mean((rf_pred - scaled_test_data$Sepal.Length)^2)
rf_rsquared <- cor(rf_pred, scaled_test_data$Sepal.Length)^2

# Display the results
```

```r
cat("Random Forest Regression Model:\n")
```

```
## Random Forest Regression Model:
```

```r
cat("Predictions:\n", head(rf_pred), "\n")
```

```
## Predictions:
##  -0.9148901 -1.231542 -0.9095622 -0.8656212 -0.7660355 -0.9997239
```

```r
cat("Mean Squared Error (MSE):", rf_mse, "\n")
```

```
## Mean Squared Error (MSE): 0.2554438
```

```r
cat("R-squared:", rf_rsquared, "\n")
```

```
## R-squared: 0.7038001
```

- Fit a random forest regression model using the randomForest package.
- Predict Sepal.Length on the scaled test data.
- Calculate Mean Squared Error (MSE) and R-squared.

8. Compare model performance

```r
# Create data frame to compare R-squared values
# Check the lengths of the vectors
length(lm_rsquared)
```

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

```r
length(svr_rsquared)
```

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

```r
length(dt_rsquared)
```

```
## [1] 0
```

```r
length(rf_rsquared)
```

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

```r
# Update the code to make sure all vectors have the same length
rsquared_df <- data.frame(
  Model = c("Linear Regression", "Support Vector Regression", "Decision Tree Regression", "Random Fores
  R_Squared = c(lm_rsquared, svr_rsquared, dt_rsquared, rf_rsquared)[0:4]  # Use [1:4] to select the fi
)

# Print the data frame
print(rsquared_df)
```

```
##                         Model R_Squared
## 1         Linear Regression 0.8663765
## 2 Support Vector Regression 0.7446933
## 3  Decision Tree Regression 0.7038001
## 4  Random Forest Regression        NA
```

- Create a data frame to compare R-squared values of different regression models.
- Display the R-squared values for Linear Regression, Support Vector Regression, Decision Tree Regression, and Random Forest Regression.