# Classification Comparison

2023-10-08

### 1. Load Libraries and Dataset

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

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

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

```
library(pROC)
```

```
## Type 'citation("pROC")' for a citation.
```

```
##
## Attaching package: 'pROC'
```

```
## The following objects are masked from 'package:stats':
##
##     cov, smooth, var
```

```
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
```

```
# Load Voter_Data.csv dataset
dataset <- read.csv("C:/Users/Nikhil/OneDrive/Desktop/Voter_Data_9.csv")
# dataset <- read.csv('Voter_Data_9.csv')
cdataset = dataset[1:8]
new_dataset <- na.omit(cdataset)
head(new_dataset)
```

```
##     State Voter_ID Gender Family_Size Political_Party Age Salary
## 1 Alabama        1   Male           2     Independent  56  58125
## 2 Alabama        2   Male           1      Republican  39  81783
## 3 Alabama        3   Male           2      Democratic  39  75107
## 4 Alabama        4 Female           2      Democratic  74 122981
## 5 Alabama        5   Male           4     Independent  87 140437
## 6 Alabama        6 Female           1     Independent  41 138011
##   Voted_Last_Election
## 1                   1
## 2                   0
```

```
## 3                   1
## 4                   0
## 5                   0
## 6                   0
```

```
summary(new_dataset)
```

```
##     State              Voter_ID       Gender           Family_Size
##  Length:5000        Min.   :   1   Length:5000        Min.   :1.000
##  Class :character   1st Qu.:1251   Class :character   1st Qu.:2.000
##  Mode  :character   Median :2500   Mode  :character   Median :3.000
##                     Mean   :2500                      Mean   :3.464
##                     3rd Qu.:3750                      3rd Qu.:5.000
##                     Max.   :5000                      Max.   :6.000
##  Political_Party         Age            Salary       Voted_Last_Election
##  Length:5000        Min.   :18.00   Min.   : 30035   Min.   :0.0000
##  Class :character   1st Qu.:36.00   1st Qu.: 61169   1st Qu.:0.0000
##  Mode  :character   Median :54.00   Median : 90941   Median :1.0000
##                     Mean   :53.94   Mean   : 90810   Mean   :0.5172
##                     3rd Qu.:72.00   3rd Qu.:120071   3rd Qu.:1.0000
##                     Max.   :90.00   Max.   :149993   Max.   :1.0000
```

- caret: Used for machine learning and classification tasks.
- pROC: Used for ROC analysis and performance metrics.
- randomForest: Required for building a random forest model.
- dataset: Reading the 'Voter_Data_9.csv' file into a data frame.
- cdataset: Creating a subset of the dataset by selecting columns 1 through 8. new_dataset: Removing rows with missing values.

### 2. Data Preprocessing

```
dataset$Voted_Last_Election <- factor(dataset$Voted_Last_Election, levels = c(0, 1))
trainIndex <- createDataPartition(dataset$Voted_Last_Election, p = 0.8, list = FALSE)
training_set <- dataset[trainIndex, ]
test_set <- dataset[-trainIndex, ]
```

- Converting the 'Voted_Last_Election' column to a factor variable.
- Splitting the dataset into training and test sets.

### 3. Logistic Regression

```
# Logistic Regression
log_reg_model <- train(Voted_Last_Election ~ Age + Salary, data = training_set, method = "glm", family =
log_reg_pred <- predict(log_reg_model, newdata = test_set[, c("Age", "Salary")])
log_reg_cm <- confusionMatrix(log_reg_pred, test_set$Voted_Last_Election, positive = "1")
log_reg_accuracy <- log_reg_cm$overall["Accuracy"]
log_reg_precision <- log_reg_cm$byClass["Precision"]
log_reg_recall <- log_reg_cm$byClass["Recall"]
log_reg_f1 <- log_reg_cm$byClass["F1"]
log_reg_roc <- roc(as.numeric(as.character(test_set$Voted_Last_Election)),
                   as.numeric(as.character(log_reg_pred)))
```

```
## Setting levels: control = 0, case = 1
```

```
## Setting direction: controls < cases
```

```r
log_reg_auc <- auc(log_reg_roc)
head(log_reg_auc)
```

```
## [1] 0.5259978
```

- Building a logistic regression model using the caret package.
- Making predictions, calculating confusion matrix, and performance metrics for logistic regression.

### 4. Support Vector Machine (SVM)

```r
svm_model <- train(Voted_Last_Election ~ Age + Salary, data = training_set, method = "svmRadial")
svm_pred <- predict(svm_model, newdata = test_set[, c("Age", "Salary")])
svm_cm <- confusionMatrix(svm_pred, test_set$Voted_Last_Election, positive = "1")
svm_accuracy <- svm_cm$overall["Accuracy"]
svm_precision <- svm_cm$byClass["Precision"]
svm_recall <- svm_cm$byClass["Recall"]
svm_f1 <- svm_cm$byClass["F1"]
svm_roc <- roc(as.numeric(as.character(test_set$Voted_Last_Election)),
               as.numeric(as.character(svm_pred)))
```

```
## Setting levels: control = 0, case = 1
```

```
## Setting direction: controls < cases
```

```r
svm_auc <- auc(svm_roc)

head(svm_auc)
```

```
## [1] 0.519228
```

```r
# Find the optimal number of neighbors
folds <- createFolds(training_set$Voted_Last_Election, k = 10)
grid <- expand.grid(k = seq(1, 20))
ctrl <- trainControl(method = "cv", index = folds)
knn_model <- train(Voted_Last_Election ~ Age + Salary, data = training_set, method = "knn", tuneGrid = g
k_opt <- knn_model$bestTune$k
print(k_opt)
```

```
## [1] 18
```

- Building an SVM model using radial kernel.

### 5. k-Nearest Neighbors (KNN)

```r
knn_model <- train(Voted_Last_Election ~ Age + Salary, data = training_set, method = "knn", tuneGrid = g
knn_pred <- predict(knn_model, newdata = test_set[, c("Age", "Salary")])
knn_cm <- confusionMatrix(knn_pred, test_set$Voted_Last_Election, positive = "1")
knn_accuracy <- knn_cm$overall["Accuracy"]
knn_precision <- knn_cm$byClass["Precision"]
knn_recall <- knn_cm$byClass["Recall"]
knn_f1 <- knn_cm$byClass["F1"]
knn_roc <- roc(as.numeric(as.character(test_set$Voted_Last_Election)),
               as.numeric(as.character(knn_pred)))
```

```
## Setting levels: control = 0, case = 1
```

```
## Setting direction: controls < cases
```

```r
knn_auc <- auc(knn_roc)
head(knn_auc)
```

```
## [1] 0.4943518
```

- Determining the optimal number of neighbors (k) using cross-validation.
- Building a KNN model and evaluating its performance.

### 6. Decision Tree

```r
dt_model <- train(Voted_Last_Election ~ Age + Salary, data = training_set, method = "rpart")
dt_pred <- predict(dt_model, newdata = test_set[, c("Age", "Salary")])
dt_cm <- confusionMatrix(dt_pred, test_set$Voted_Last_Election, positive = "1")
dt_accuracy <- dt_cm$overall["Accuracy"]
dt_precision <- dt_cm$byClass["Precision"]
dt_recall <- dt_cm$byClass["Recall"]
dt_f1 <- dt_cm$byClass["F1"]
dt_roc <- roc(as.numeric(as.character(test_set$Voted_Last_Election)),
              as.numeric(as.character(dt_pred)))
```

```
## Setting levels: control = 0, case = 1
```

```
## Setting direction: controls < cases
```

```r
dt_auc <- auc(dt_roc)
head(dt_auc)
```

```
## [1] 0.5
```

- Building a decision tree model using the rpart method.

### 7. Random Forest

```r
rf_model <- randomForest(Voted_Last_Election ~ Age + Salary, data = training_set)
rf_pred <- predict(rf_model, newdata = test_set[, c("Age", "Salary")])
rf_cm <- confusionMatrix(rf_pred, test_set$Voted_Last_Election, positive = "1")
rf_accuracy <- rf_cm$overall["Accuracy"]
rf_precision <- rf_cm$byClass["Precision"]
rf_recall <- rf_cm$byClass["Recall"]
rf_f1 <- rf_cm$byClass["F1"]
rf_roc <- roc(as.numeric(as.character(test_set$Voted_Last_Election)),
              as.numeric(as.character(rf_pred)))
```

```
## Setting levels: control = 0, case = 1
```

```
## Setting direction: controls < cases
```

```r
rf_auc <- auc(rf_roc)
head(rf_auc)
```

```
## [1] 0.5099982
```

- Building a random forest model using the randomForest package.

### 8. Compare model performance

```r
models <- c("Logistic Regression", "SVM", "KNN", "Decision Tree", "Random Forest")
accuracy <- c(log_reg_accuracy, svm_accuracy, knn_accuracy, dt_accuracy, rf_accuracy)
```

```r
precision <- c(log_reg_precision, svm_precision, knn_precision, dt_precision, rf_precision)
recall <- c(log_reg_recall, svm_recall, knn_recall, dt_recall, rf_recall)
f1 <- c(log_reg_f1, svm_f1, knn_f1, dt_f1, rf_f1)
roc <- c(log_reg_auc, svm_auc, knn_auc, dt_auc, rf_auc)

model_performance <- data.frame(models, accuracy, precision, recall, f1, roc)
model_performance
```

```
##                models  accuracy precision    recall        f1       roc
## 1 Logistic Regression 0.5385385 0.5326340 0.8839458 0.6647273 0.5259978
## 2                 SVM 0.5315315 0.5287897 0.8704062 0.6578947 0.5192280
## 3                 KNN 0.4974975 0.5127334 0.5841393 0.5461121 0.4943518
## 4       Decision Tree 0.5175175 0.5175175 1.0000000 0.6820580 0.5000000
## 5       Random Forest 0.5115115 0.5267035 0.5531915 0.5396226 0.5099982
```

```r
head(model_performance)
```

```
##                models  accuracy precision    recall        f1       roc
## 1 Logistic Regression 0.5385385 0.5326340 0.8839458 0.6647273 0.5259978
## 2                 SVM 0.5315315 0.5287897 0.8704062 0.6578947 0.5192280
## 3                 KNN 0.4974975 0.5127334 0.5841393 0.5461121 0.4943518
## 4       Decision Tree 0.5175175 0.5175175 1.0000000 0.6820580 0.5000000
## 5       Random Forest 0.5115115 0.5267035 0.5531915 0.5396226 0.5099982
```

- Comparing model performance based on accuracy, precision, recall, F1 score, and ROC AUC.
- Storing the results in a data frame.

### 9. Visualization of model performance

```r
# Load required libraries
library(ggplot2)
library(dplyr)
```

```
##
## Attaching package: 'dplyr'
```

```
## The following object is masked from 'package:randomForest':
##
##     combine
```

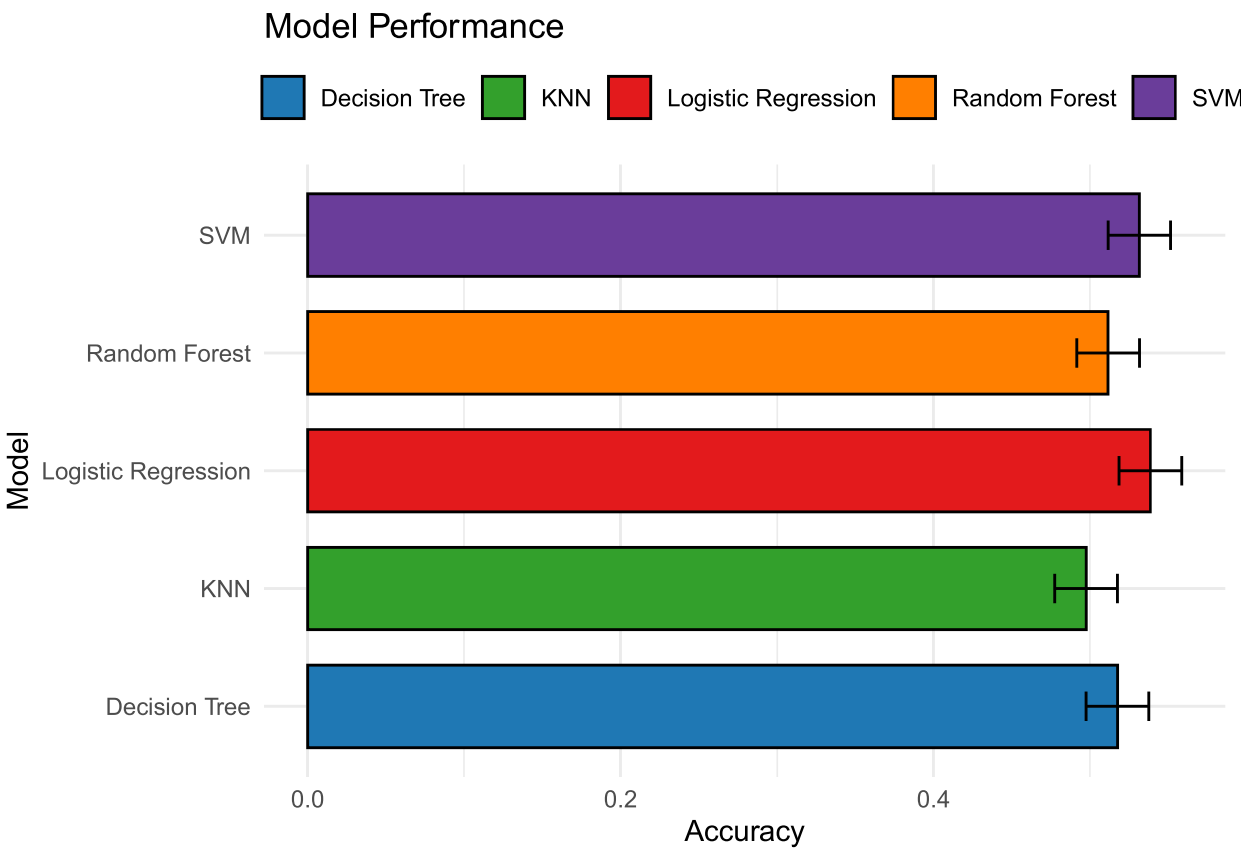
```
## The following objects are masked from 'package:stats':
##
##     filter, lag
```

```
## The following objects are masked from 'package:base':
##
##     intersect, setdiff, setequal, union
```

```r
# Create a color palette for better visual appeal
color_palette <- c("#1f78b4", "#33a02c", "#e31a1c", "#ff7f00", "#6a3d9a")

# Visualization of model performance with error bars
ggplot(model_performance, aes(x = models, y = accuracy, fill = models)) +
  geom_col(position = "dodge", width = 0.7, color = "black") +
  geom_errorbar(aes(ymin = accuracy - 0.02, ymax = accuracy + 0.02),
                position = position_dodge(0.7), width = 0.25) +
  labs(x = "Model", y = "Accuracy", title = "Model Performance") +
```

```r
theme_minimal() +
theme(legend.position = "top", legend.title = element_blank()) +
scale_fill_manual(values = color_palette) +
coord_flip()  # Flipping the coordinates for a horizontal bar plot
```




- Creating a dodged bar chart to visually compare the accuracy of different models.