Model Comparison

Enoch

2025-08-01

Model Comparisons

Final Year Undergraduate Project

Load Packages

```
library(caret)
```

```
## Warning: package 'caret' was built under R version 4.4.2
```

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

```
## Warning: package 'ggplot2' was built under R version 4.4.3
```

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

```
library(e1071)       # For Naive Bayes
```

```
## Warning: package 'e1071' was built under R version 4.4.2
```

```
library(randomForest)
```

```
## Warning: package 'randomForest' was built under R version 4.4.2
```

```
## randomForest 4.7-1.2
```

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

```
##
## Attaching package: 'randomForest'
```

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

```r
library(rpart)       # For Decision Tree
```

```
## Warning: package 'rpart' was built under R version 4.4.2
```

```r
library(MLmetrics)
```

```
## Warning: package 'MLmetrics' was built under R version 4.4.2
```

```
##
## Attaching package: 'MLmetrics'
```

```
## The following objects are masked from 'package:caret':
##
##     MAE, RMSE
```

```
## The following object is masked from 'package:base':
##
##     Recall
```

```r
library(data.table)
```

```
## Warning: package 'data.table' was built under R version 4.4.2
```

```r
library(ggplot2)
library(class)
```

```
## Warning: package 'class' was built under R version 4.4.2
```

```r
library(pROC)
```

```
## Warning: package 'pROC' was built under R version 4.4.2
```

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

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

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

```r
library(MASS)
```

```r
# Function to generate synthetic data
generate_data <- function(n) {
  set.seed(123)
  p <- 10  # Number of predictors

  # Generate multivariate normal features
  mean_vector <- rep(0, p)
  covariance_matrix <- diag(p)
  X <- mvrnorm(n, mu = mean_vector, Sigma = covariance_matrix)

  # Convert to data frame
  df <- as.data.frame(X)
  colnames(df) <- paste0("x", 1:p)

  # Define logistic function
  logistic <- function(z) { 1 / (1 + exp(-z)) }

  # Define coefficients and intercept
  beta <- runif(p, -0.5, 0.5)
  intercept <- 0

  # Compute linear predictor and probabilities
  logit_values <- as.matrix(df) %*% beta + intercept
  probabilities <- logistic(logit_values)

  # Generate binary response variable
  df$y <- factor(rbinom(n, 1, probabilities), levels = c(0, 1))

  return(df)
}

# Function to calculate metrics
calculate_metrics <- function(actual, predicted, prob) {
  actual <- factor(actual, levels = c("0", "1"))
  predicted <- factor(predicted, levels = c("0", "1"))

  # Confusion Matrix
  confusion <- confusionMatrix(predicted, actual, positive = "1")

  # Precision, Recall, and F1 Score
  precision <- confusion$byClass["Precision"]
  recall <- confusion$byClass["Recall"]
  f1 <- (2 * precision * recall) / (precision + recall)  # F1 Score

  # AUC Calculation
  auc_value <- auc(roc(actual, prob))

  return(data.frame(Precision = precision, Recall = recall, F1Score = f1, AUC = auc_value))
}

# Define sample sizes
sample_sizes <- c(100, 500, 1000, 2000)
results <- list()
```

```r
# Iterate over different sample sizes
for (n in sample_sizes) {
  set.seed(123)

  # Generate data
  data <- generate_data(n)
  train_index <- createDataPartition(data$y, p = 0.8, list = FALSE)
  train <- data[train_index, ]
  test <- data[-train_index, ]

  # Logistic Regression
  log_model <- glm(y ~ ., family = binomial, data = train)
  log_prob <- predict(log_model, test, type = "response")
  log_pred <- factor(ifelse(log_prob > 0.5, "1", "0"), levels = c("0", "1"))

  # K-Nearest Neighbors
  knn_pred <- knn(train = train[, -ncol(train)], test = test[, -ncol(test)], cl = train$y, k = 5)
  knn_prob <- as.numeric(knn_pred) - 1  # Convert factor to numeric probability

  # Naive Bayes
  nb_model <- naiveBayes(y ~ ., data = train)
  nb_prob <- predict(nb_model, test, type = "raw")[, 2]
  nb_pred <- predict(nb_model, test)

  # Decision Tree
  dt_model <- rpart(y ~ ., data = train, method = "class")
  dt_prob <- predict(dt_model, test, type = "prob")[, 2]
  dt_pred <- predict(dt_model, test, type = "class")

  # Random Forest
  rf_model <- randomForest(y ~ ., data = train, ntree = 100)
  rf_prob <- predict(rf_model, test, type = "prob")[, 2]
  rf_pred <- predict(rf_model, test, type = "class")

  # Compute metrics for all models
  log_metrics <- calculate_metrics(test$y, log_pred, log_prob)
  knn_metrics <- calculate_metrics(test$y, knn_pred, knn_prob)
  nb_metrics <- calculate_metrics(test$y, nb_pred, nb_prob)
  dt_metrics <- calculate_metrics(test$y, dt_pred, dt_prob)
  rf_metrics <- calculate_metrics(test$y, rf_pred, rf_prob)

  # Store results
  results[[paste0("n_", n)]] <- rbind(
    data.table(Model = "Logistic Regression", SampleSize = n, Precision = log_metrics$Precision, Recall
    data.table(Model = "KNN", SampleSize = n, Precision = knn_metrics$Precision, Recall = knn_metrics$R
    data.table(Model = "Naive Bayes", SampleSize = n, Precision = nb_metrics$Precision, Recall = nb_met
    data.table(Model = "Decision Tree", SampleSize = n, Precision = dt_metrics$Precision, Recall = dt_m
    data.table(Model = "Random Forest", SampleSize = n, Precision = rf_metrics$Precision, Recall = rf_m
  )
}
```

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

```
## Setting direction: controls < cases

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

## Setting direction: controls > cases

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

## Setting direction: controls < cases

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

## Setting direction: controls < cases

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

## Setting direction: controls > cases

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

## Setting direction: controls < cases

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

## Setting direction: controls < cases

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

## Setting direction: controls < cases

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

## Setting direction: controls < cases

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

## Setting direction: controls < cases

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

## Setting direction: controls < cases

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

## Setting direction: controls < cases

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

```
## Setting direction: controls < cases

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

## Setting direction: controls < cases

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

## Setting direction: controls < cases

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

## Setting direction: controls < cases

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

## Setting direction: controls < cases

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

## Setting direction: controls < cases

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

## Setting direction: controls < cases

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

## Setting direction: controls < cases
```

Combining Result

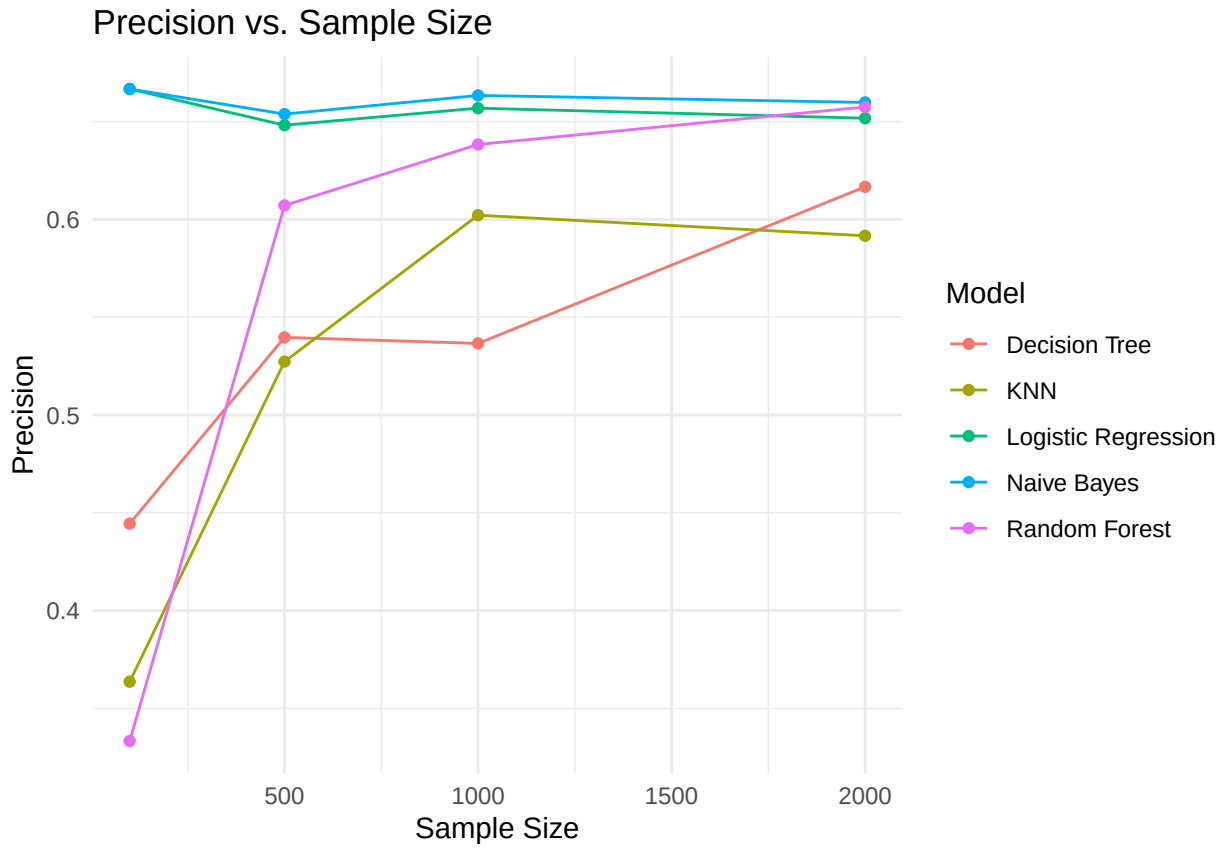
```r
# Combine all results
final_results <- rbindlist(results)
print(final_results)
```

```
##                   Model SampleSize Precision    Recall   F1Score       AUC
##                  <char>      <num>     <num>     <num>     <num>     <auc>
##  1: Logistic Regression        100 0.6666667 0.5000000 0.5714286 0.6704545
##  2:                 KNN        100 0.3636364 0.5000000 0.4210526 0.5681818
##  3:         Naive Bayes        100 0.6666667 0.5000000 0.5714286 0.6477273
##  4:       Decision Tree        100 0.4444444 0.5000000 0.4705882 0.5227273
##  5:       Random Forest        100 0.3333333 0.1250000 0.1818182 0.5170455
##  6: Logistic Regression        500 0.6481481 0.6862745 0.6666667 0.7528595
##  7:                 KNN        500 0.5272727 0.5686275 0.5471698 0.5134804
##  8:         Naive Bayes        500 0.6538462 0.6666667 0.6601942 0.7549020
##  9:       Decision Tree        500 0.5396825 0.6666667 0.5964912 0.5625000
## 10:       Random Forest        500 0.6071429 0.6666667 0.6355140 0.6672794
```

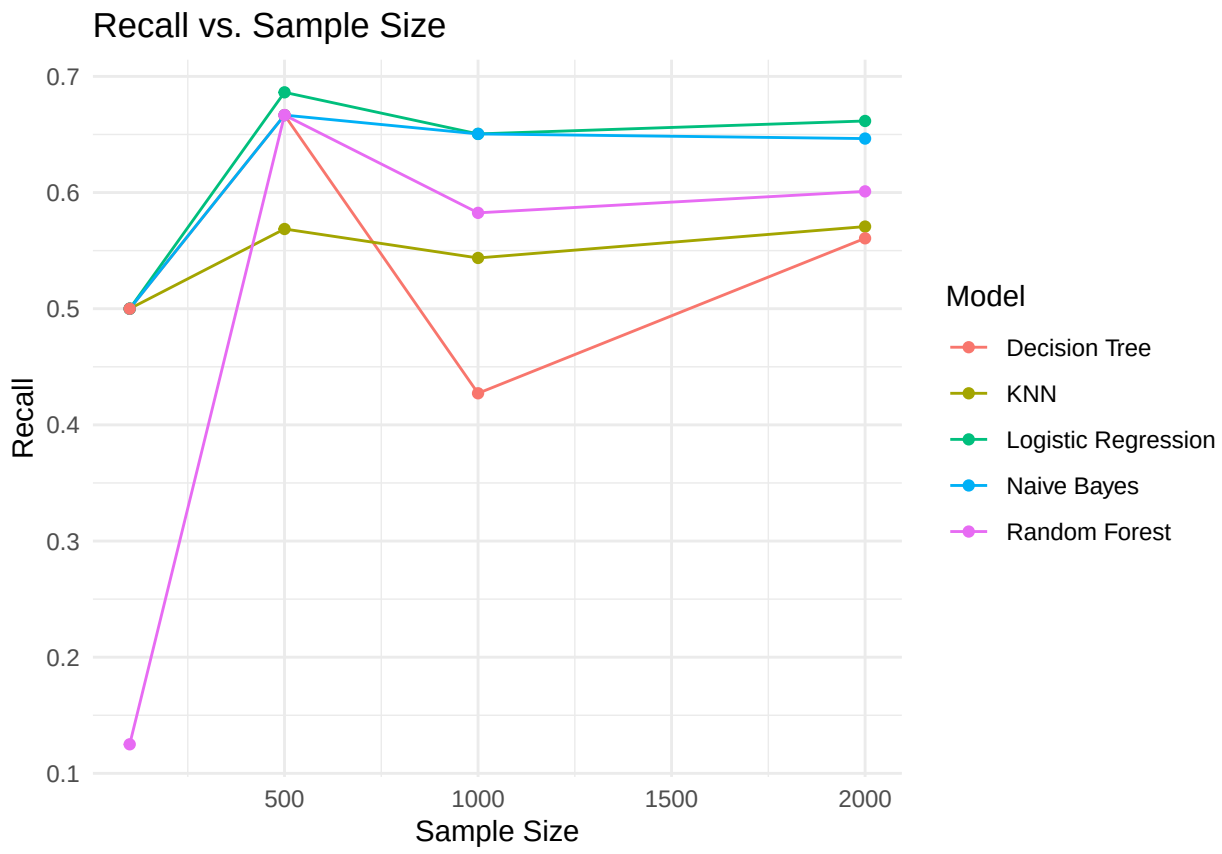
```
## 11: Logistic Regression       1000 0.6568627 0.6504854 0.6536585 0.6627225
## 12:                 KNN       1000 0.6021505 0.5436893 0.5714286 0.5791363
## 13:         Naive Bayes       1000 0.6633663 0.6504854 0.6568627 0.6695995
## 14:       Decision Tree       1000 0.5365854 0.4271845 0.4756757 0.4721885
## 15:       Random Forest       1000 0.6382979 0.5825243 0.6091371 0.6787520
## 16: Logistic Regression       2000 0.6517413 0.6616162 0.6566416 0.7306231
## 17:                 KNN       2000 0.5916230 0.5707071 0.5809769 0.5922842
## 18:         Naive Bayes       2000 0.6597938 0.6464646 0.6530612 0.7320482
## 19:       Decision Tree       2000 0.6166667 0.5606061 0.5873016 0.6434268
## 20:       Random Forest       2000 0.6574586 0.6010101 0.6279683 0.6893314
##                   Model SampleSize Precision    Recall   F1Score       AUC
```

Visualization

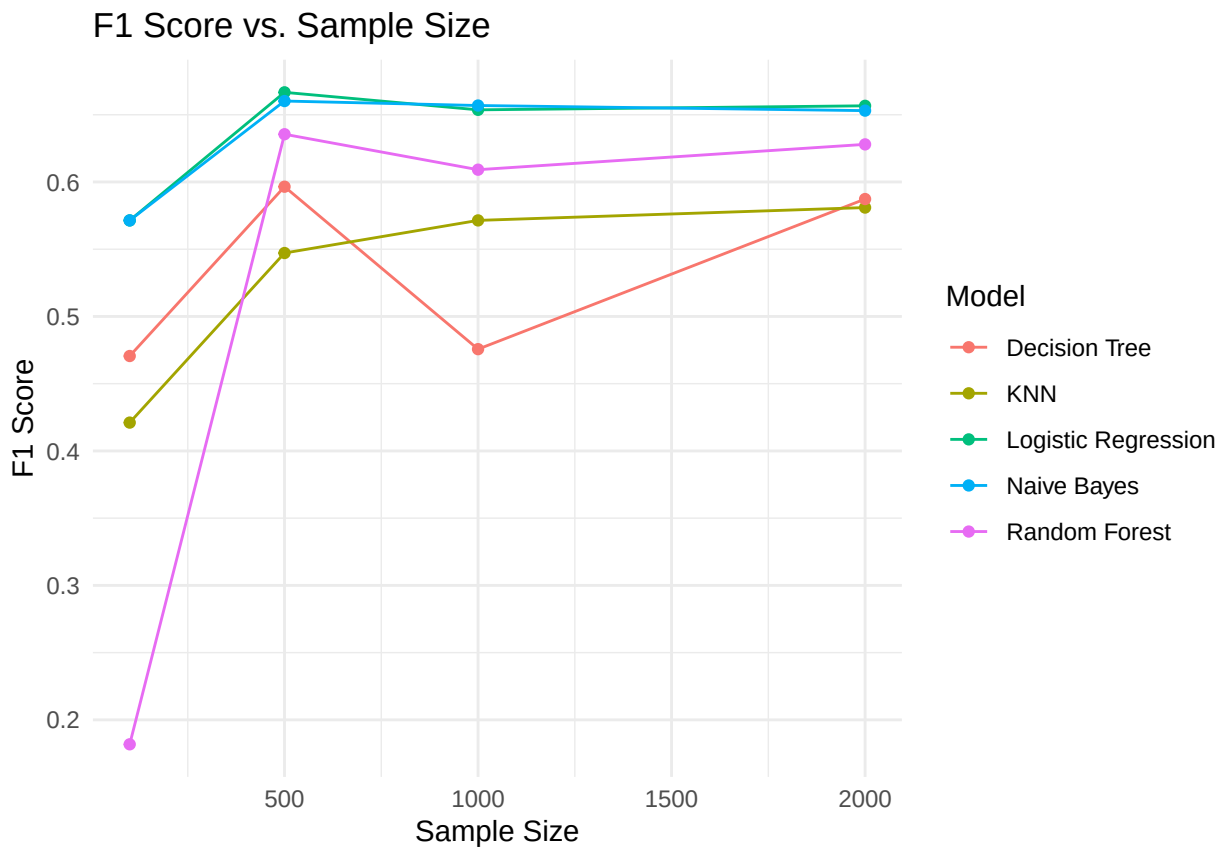
```r
# Visualization of results
ggplot(final_results, aes(x = SampleSize, y = Precision, color = Model)) +
  geom_line() + geom_point() +
  labs(title = "Precision vs. Sample Size", x = "Sample Size", y = "Precision") +
  theme_minimal()
```



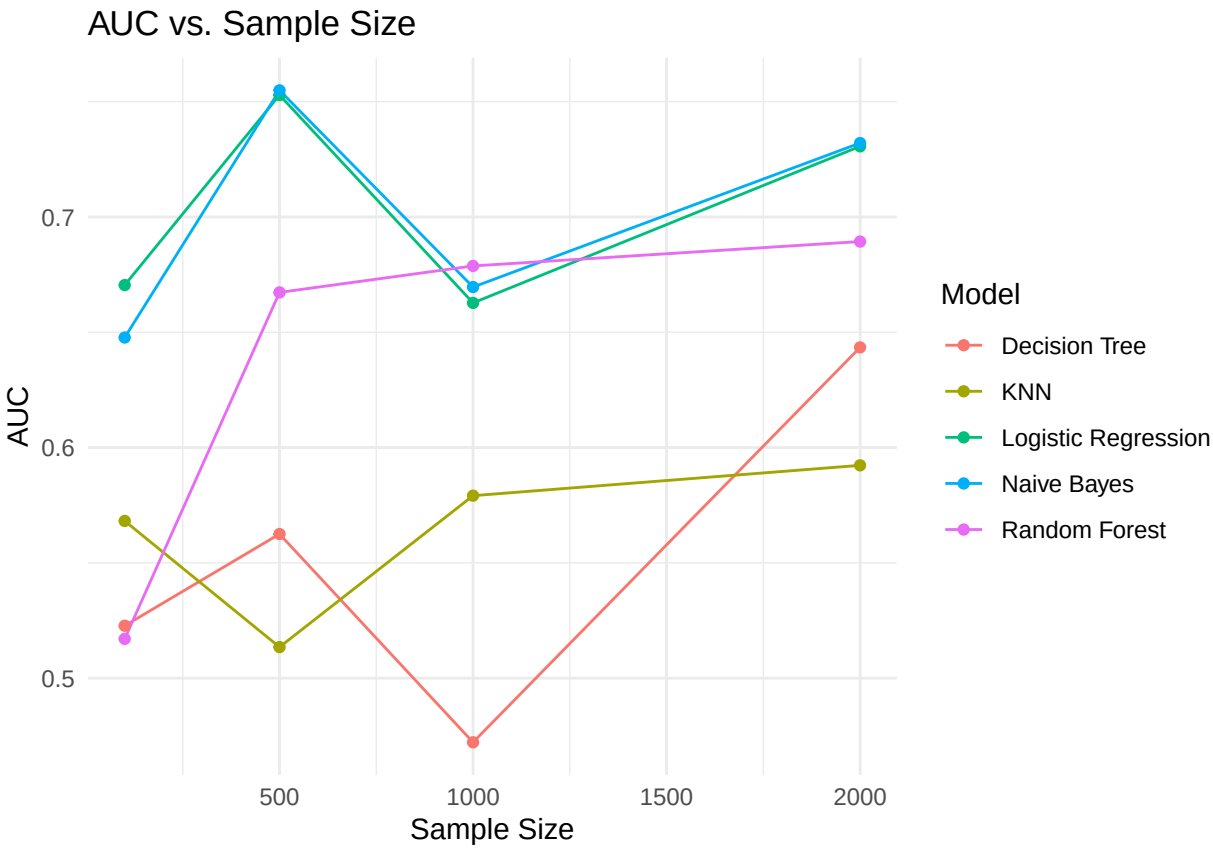

```r
ggplot(final_results, aes(x = SampleSize, y = Recall, color = Model)) +
  geom_line() + geom_point() +
  labs(title = "Recall vs. Sample Size", x = "Sample Size", y = "Recall") +
  theme_minimal()
```




```
ggplot(final_results, aes(x = SampleSize, y = F1Score, color = Model)) +
  geom_line() + geom_point() +
  labs(title = "F1 Score vs. Sample Size", x = "Sample Size", y = "F1 Score") +
  theme_minimal()
```




```
ggplot(final_results, aes(x = SampleSize, y = AUC, color = Model)) +
  geom_line() + geom_point() +
  labs(title = "AUC vs. Sample Size", x = "Sample Size", y = "AUC") +
  theme_minimal()
```


AUC vs. Sample Size
AUC
0.7
0.6
0.5
500
1000
1500
2000
Sample Size
Model
Decision Tree
KNN
Logistic Regression
Naive Bayes
Random Forest