Classification_Diabetes_Algorithms.R

Alex

2024-11-16

```
# Group 3 : Alex Hunt, Oluchi Ejehu, Zainab Iyiola
# Supervised Classification Algorithms for Early Detection of Diabetes
# Professor Nicholson : DSA-5103-995 : Final Project FA 2024
# R script that tests classification algorithms on a diabetes dataset

# Load required libraries
library(caret)
```

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

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

```
library(randomForest)
```

```
## 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
```

```
library(e1071)
library(gbm)
```

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

```
## Loaded gbm 2.2.2
```

```
## This version of gbm is no longer under development. Consider transitioning to gbm3, https://github.c
```

```
library(ggplot2)
library(reshape2)
library(pROC)
```

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

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

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

```
library(tidyr)
```

```
##
## Attaching package: 'tidyr'

## The following object is masked from 'package:reshape2':
##
##     smiths
```

```
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
```

```
# Step 1: Load the datasets
# Load original and synthetic datasets
diabetes_original <- read.csv("Diabetes_Dataset.csv", header = TRUE)
load("synthetic_density_data.RData")  # Assuming the synthetic dataset is saved in an RData file

# Step 2: Preprocess the Original Dataset
# Replace zeros with NA for specific columns in the original dataset
cols_with_zeros <- c("Glucose", "BloodPressure", "SkinThickness", "Insulin", "BMI")
diabetes_original[cols_with_zeros] <- lapply(diabetes_original[cols_with_zeros], function(x) replace(x,

# Drop rows with missing values from the original dataset
diabetes_original <- na.omit(diabetes_original)

# Remove irrelevant columns from synthetic dataset
synthetic_density_data <- synthetic_density_data %>%
  select(-ends_with("_Imputed"), -ends_with("_imp"))

# Step 3: Define Target Variable and Features
target_variable <- "Outcome"  # The column to predict
feature_columns <- setdiff(names(diabetes_original), target_variable)

# Step 4: Split Data into Training and Test Sets
set.seed(123)  # For reproducibility
train_index_original <- createDataPartition(diabetes_original[[target_variable]], p = 0.7, list = FALSE
train_data_original <- diabetes_original[train_index_original, ]
test_data_original <- diabetes_original[-train_index_original, ]

train_index_synthetic <- createDataPartition(synthetic_density_data[[target_variable]], p = 0.7, list =
train_data_synthetic <- synthetic_density_data[train_index_synthetic, ]
test_data_synthetic <- synthetic_density_data[-train_index_synthetic, ]

# Step 5: Train and Evaluate Models
```

```r
# Define a function to train and evaluate models
train_and_evaluate <- function(train_data, test_data, model_name) {
  results <- list()

  # Logistic Regression
  if (model_name == "Logistic Regression") {
    model <- glm(Outcome ~ ., data = train_data, family = binomial)
    predictions <- ifelse(predict(model, test_data, type = "response") > 0.5, 1, 0)
  }

  # Random Forest
  if (model_name == "Random Forest") {
    model <- randomForest(as.factor(Outcome) ~ ., data = train_data, ntree = 100)
    predictions <- predict(model, test_data)
  }

  # SVM
  if (model_name == "SVM") {
    model <- svm(as.factor(Outcome) ~ ., data = train_data, kernel = "linear", probability = TRUE)
    predictions <- predict(model, test_data)
  }

  # Gradient Boosting
  if (model_name == "Gradient Boosting") {
    model <- gbm(Outcome ~ ., data = train_data, distribution = "bernoulli", n.trees = 100, interaction
    predictions <- ifelse(predict(model, test_data, n.trees = 100, type = "response") > 0.5, 1, 0)
  }

  # Confusion Matrix and Accuracy
  confusion <- confusionMatrix(as.factor(predictions), as.factor(test_data$Outcome))
  results$model <- model
  results$confusion <- confusion
  return(results)
}

# List of models to evaluate
models <- c("Logistic Regression", "Random Forest", "SVM", "Gradient Boosting")

# Train models on Original Dataset
cat("Results on Original Dataset:\n")
```

```
## Results on Original Dataset:
```

```r
original_results <- lapply(models, function(m) {
  cat("\nModel:", m, "\n")
  result <- train_and_evaluate(train_data_original, test_data_original, m)
  print(result$confusion)
  return(result)
})
```

```
##
## Model: Logistic Regression
## Confusion Matrix and Statistics
##
##           Reference
```

```
## Prediction  0  1
##          0 74 14
##          1  8 21
##
##                Accuracy : 0.812
##                  95% CI : (0.7293, 0.8782)
##     No Information Rate : 0.7009
##     P-Value [Acc > NIR] : 0.004437
##
##                   Kappa : 0.5284
##
##  Mcnemar's Test P-Value : 0.286422
##
##             Sensitivity : 0.9024
##             Specificity : 0.6000
##          Pos Pred Value : 0.8409
##          Neg Pred Value : 0.7241
##              Prevalence : 0.7009
##          Detection Rate : 0.6325
##    Detection Prevalence : 0.7521
##       Balanced Accuracy : 0.7512
##
##        'Positive' Class : 0
##
##
## Model: Random Forest
## Confusion Matrix and Statistics
##
##           Reference
## Prediction  0  1
##          0 69 17
##          1 13 18
##
##                Accuracy : 0.7436
##                  95% CI : (0.6546, 0.8198)
##     No Information Rate : 0.7009
##     P-Value [Acc > NIR] : 0.1824
##
##                   Kappa : 0.3678
##
##  Mcnemar's Test P-Value : 0.5839
##
##             Sensitivity : 0.8415
##             Specificity : 0.5143
##          Pos Pred Value : 0.8023
##          Neg Pred Value : 0.5806
##              Prevalence : 0.7009
##          Detection Rate : 0.5897
##    Detection Prevalence : 0.7350
##       Balanced Accuracy : 0.6779
##
##        'Positive' Class : 0
##
##
```

```
## Model: SVM
## Confusion Matrix and Statistics
##
##           Reference
## Prediction  0  1
##          0 71 16
##          1 11 19
##
##                Accuracy : 0.7692
##                  95% CI : (0.6823, 0.8421)
##     No Information Rate : 0.7009
##     P-Value [Acc > NIR] : 0.06242
##
##                   Kappa : 0.4262
##
##  Mcnemar's Test P-Value : 0.44142
##
##             Sensitivity : 0.8659
##             Specificity : 0.5429
##          Pos Pred Value : 0.8161
##          Neg Pred Value : 0.6333
##              Prevalence : 0.7009
##          Detection Rate : 0.6068
##    Detection Prevalence : 0.7436
##       Balanced Accuracy : 0.7044
##
##        'Positive' Class : 0
##
##
## Model: Gradient Boosting
## Confusion Matrix and Statistics
##
##           Reference
## Prediction  0  1
##          0 70 17
##          1 12 18
##
##                Accuracy : 0.7521
##                  95% CI : (0.6638, 0.8273)
##     No Information Rate : 0.7009
##     P-Value [Acc > NIR] : 0.1326
##
##                   Kappa : 0.3837
##
##  Mcnemar's Test P-Value : 0.4576
##
##             Sensitivity : 0.8537
##             Specificity : 0.5143
##          Pos Pred Value : 0.8046
##          Neg Pred Value : 0.6000
##              Prevalence : 0.7009
##          Detection Rate : 0.5983
##    Detection Prevalence : 0.7436
##       Balanced Accuracy : 0.6840
```

```
##
##        'Positive' Class : 0
##
```

```
# Train models on Synthetic Dataset
cat("\nResults on Synthetic Dataset:\n")
```

```
##
## Results on Synthetic Dataset:
```

```
synthetic_results <- lapply(models, function(m) {
  cat("\nModel:", m, "\n")
  result <- train_and_evaluate(train_data_synthetic, test_data_original, m)
  print(result$confusion)
  return(result)
})
```

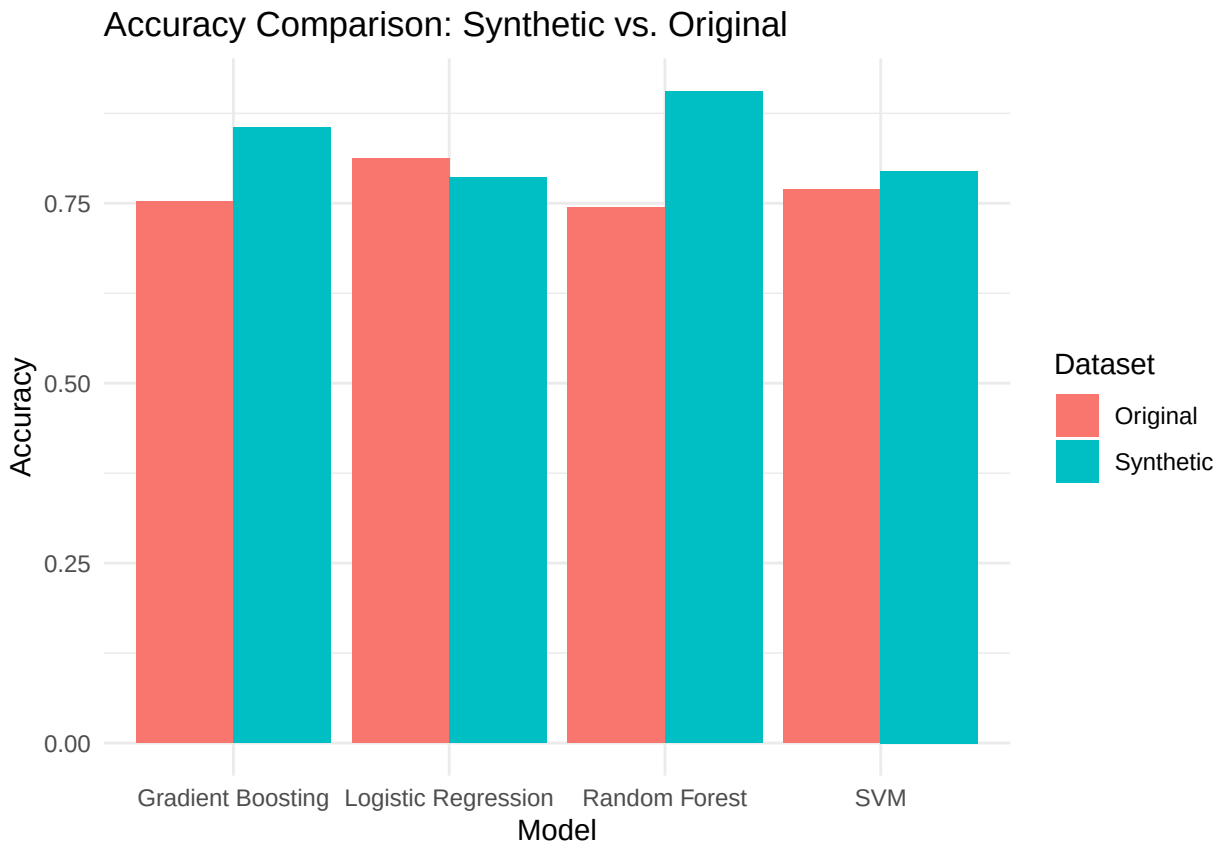
```
##
## Model: Logistic Regression
## Confusion Matrix and Statistics
##
##           Reference
## Prediction  0  1
##          0 70 13
##          1 12 22
##
##                Accuracy : 0.7863
##                  95% CI : (0.7009, 0.8567)
##     No Information Rate : 0.7009
##     P-Value [Acc > NIR] : 0.02489
##
##                   Kappa : 0.4862
##
##  Mcnemar's Test P-Value : 1.00000
##
##             Sensitivity : 0.8537
##             Specificity : 0.6286
##          Pos Pred Value : 0.8434
##          Neg Pred Value : 0.6471
##              Prevalence : 0.7009
##          Detection Rate : 0.5983
##    Detection Prevalence : 0.7094
##       Balanced Accuracy : 0.7411
##
##        'Positive' Class : 0
##
##
## Model: Random Forest
## Confusion Matrix and Statistics
##
##           Reference
## Prediction  0  1
##          0 76  5
##          1  6 30
##
##                Accuracy : 0.906
```

```
##                  95% CI : (0.838, 0.9521)
##     No Information Rate : 0.7009
##     P-Value [Acc > NIR] : 8.42e-08
##
##                   Kappa : 0.7776
##
##  Mcnemar's Test P-Value : 1
##
##             Sensitivity : 0.9268
##             Specificity : 0.8571
##          Pos Pred Value : 0.9383
##          Neg Pred Value : 0.8333
##              Prevalence : 0.7009
##          Detection Rate : 0.6496
##    Detection Prevalence : 0.6923
##       Balanced Accuracy : 0.8920
##
##        'Positive' Class : 0
##
##
## Model: SVM
## Confusion Matrix and Statistics
##
##           Reference
## Prediction  0  1
##          0 70 12
##          1 12 23
##
##                Accuracy : 0.7949
##                  95% CI : (0.7103, 0.8639)
##     No Information Rate : 0.7009
##     P-Value [Acc > NIR] : 0.01469
##
##                   Kappa : 0.5108
##
##  Mcnemar's Test P-Value : 1.00000
##
##             Sensitivity : 0.8537
##             Specificity : 0.6571
##          Pos Pred Value : 0.8537
##          Neg Pred Value : 0.6571
##              Prevalence : 0.7009
##          Detection Rate : 0.5983
##    Detection Prevalence : 0.7009
##       Balanced Accuracy : 0.7554
##
##        'Positive' Class : 0
##
##
## Model: Gradient Boosting
## Confusion Matrix and Statistics
##
##           Reference
## Prediction  0  1
```

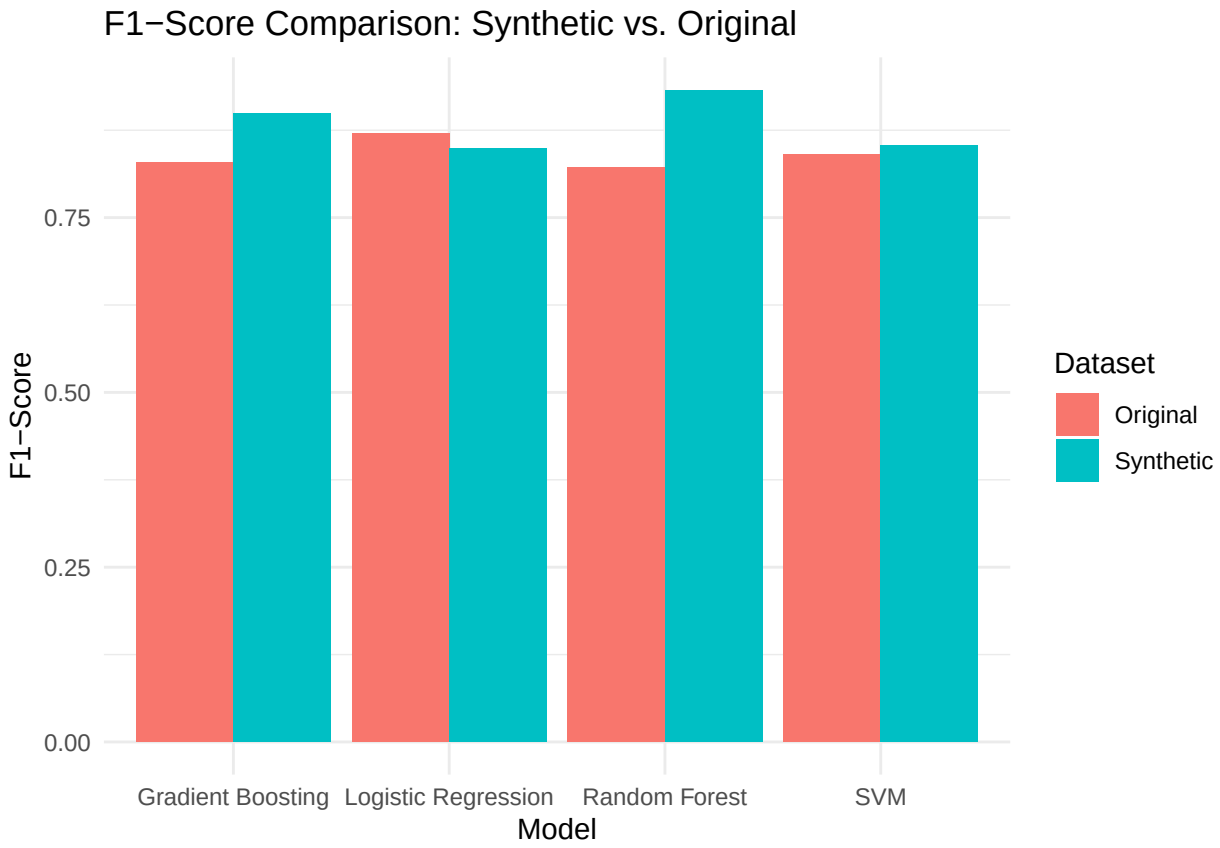
```
##          0 76 11
##          1  6 24
##
##                Accuracy : 0.8547
##                  95% CI : (0.7776, 0.913)
##     No Information Rate : 0.7009
##     P-Value [Acc > NIR] : 8.658e-05
##
##                   Kappa : 0.6387
##
##  Mcnemar's Test P-Value : 0.332
##
##             Sensitivity : 0.9268
##             Specificity : 0.6857
##          Pos Pred Value : 0.8736
##          Neg Pred Value : 0.8000
##              Prevalence : 0.7009
##          Detection Rate : 0.6496
##    Detection Prevalence : 0.7436
##       Balanced Accuracy : 0.8063
##
##        'Positive' Class : 0
##
```

```r
# Function to calculate F1-Score
calculate_f1_score <- function(confusion_matrix) {
  precision <- confusion_matrix$byClass["Pos Pred Value"]
  recall <- confusion_matrix$byClass["Sensitivity"]
  f1_score <- 2 * ((precision * recall) / (precision + recall))
  return(f1_score)
}

# Extract metrics from results
extract_metrics <- function(results, dataset_name) {
  metrics <- data.frame(
    Model = character(),
    Dataset = character(),
    Accuracy = numeric(),
    F1_Score = numeric(),
    stringsAsFactors = FALSE
  )
  for (i in seq_along(results)) {
    model_name <- models[i]
    confusion <- results[[i]]$confusion
    accuracy <- confusion$overall["Accuracy"]
    f1_score <- calculate_f1_score(confusion)
    metrics <- rbind(metrics, data.frame(
      Model = model_name,
      Dataset = dataset_name,
      Accuracy = accuracy,
      F1_Score = f1_score
    ))
  }
  return(metrics)
```

```
}

# Extract metrics for both datasets
metrics_original <- extract_metrics(original_results, "Original")
metrics_synthetic <- extract_metrics(synthetic_results, "Synthetic")

# Combine metrics into a single dataframe
all_metrics <- rbind(metrics_original, metrics_synthetic)

# Bar chart for Accuracy comparison
ggplot(all_metrics, aes(x = Model, y = Accuracy, fill = Dataset)) +
  geom_bar(stat = "identity", position = "dodge") +
  labs(title = "Accuracy Comparison: Synthetic vs. Original",
       x = "Model",
       y = "Accuracy") +
  theme_minimal()
```

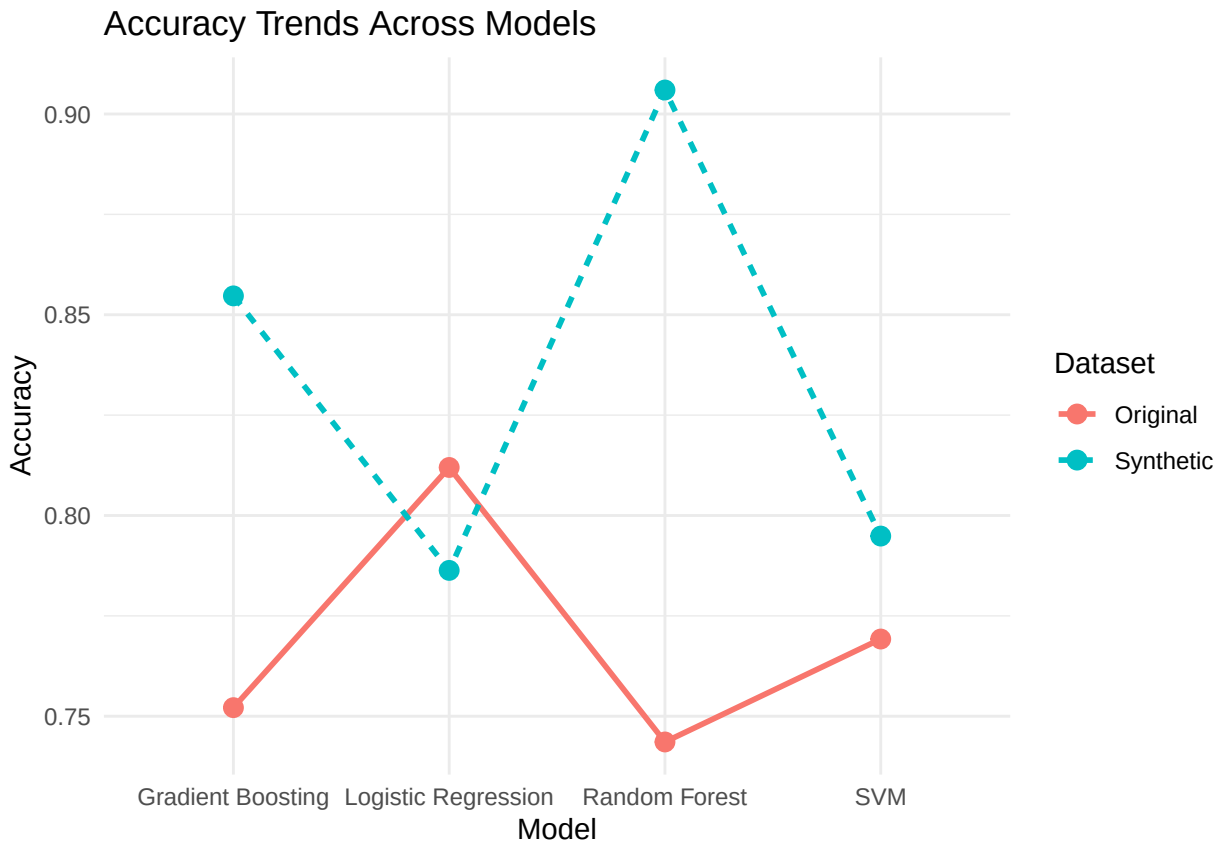



```
# Bar chart for F1-Score comparison
ggplot(all_metrics, aes(x = Model, y = F1_Score, fill = Dataset)) +
  geom_bar(stat = "identity", position = "dodge") +
  labs(title = "F1-Score Comparison: Synthetic vs. Original",
       x = "Model",
       y = "F1-Score") +
  theme_minimal()
```

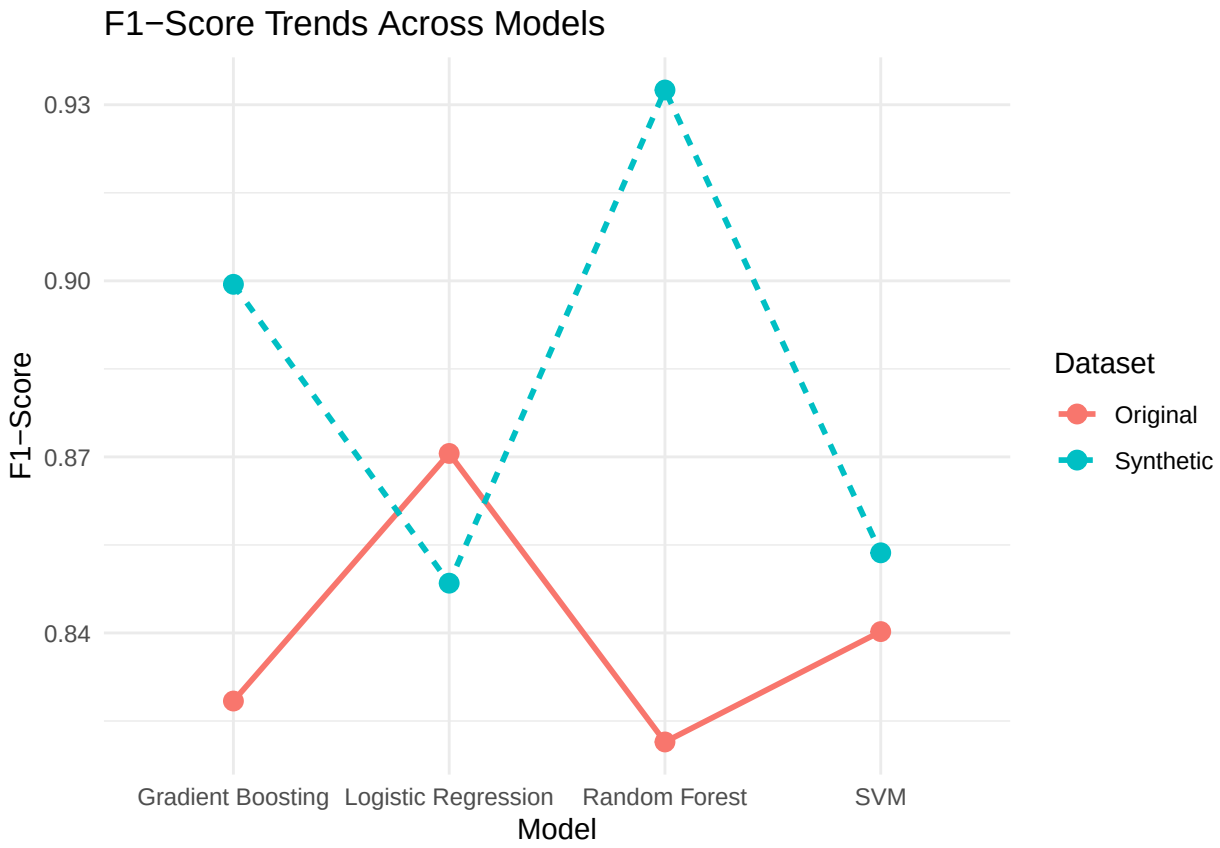



```r
# Line chart showing trends across models
ggplot(all_metrics, aes(x = Model, y = Accuracy, group = Dataset, color = Dataset)) +
  geom_line(aes(linetype = Dataset), size = 1) +
  geom_point(size = 3) +
  labs(title = "Accuracy Trends Across Models",
       x = "Model",
       y = "Accuracy") +
  theme_minimal()
```

```
## Warning: Using `size` aesthetic for lines was deprecated in ggplot2 3.4.0.
## i Please use `linewidth` instead.
## This warning is displayed once every 8 hours.
## Call `lifecycle::last_lifecycle_warnings()` to see where this warning was
## generated.
```




```
ggplot(all_metrics, aes(x = Model, y = F1_Score, group = Dataset, color = Dataset)) +
  geom_line(aes(linetype = Dataset), size = 1) +
  geom_point(size = 3) +
  labs(title = "F1-Score Trends Across Models",
       x = "Model",
       y = "F1-Score") +
  theme_minimal()
```



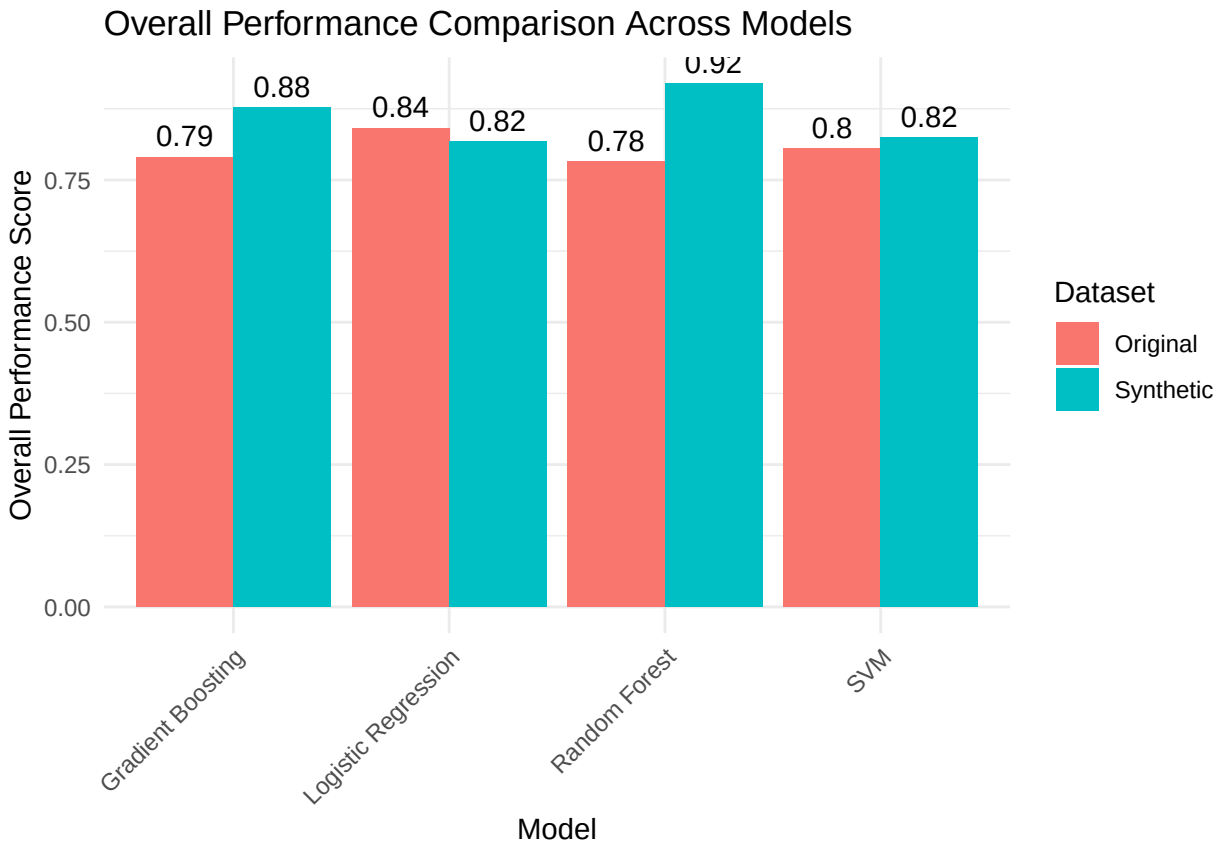

```
# Calculate Overall Performance Score
all_metrics <- all_metrics %>%
  mutate(Overall_Score = (Accuracy + F1_Score) / 2)  # Simple average of Accuracy and F1-Score

# Identify the best model
best_model <- all_metrics %>%
  arrange(desc(Overall_Score)) %>%
  slice(1)  # Select the model with the highest overall score

cat("Best Model Overall:\n")
```

```
## Best Model Overall:
```

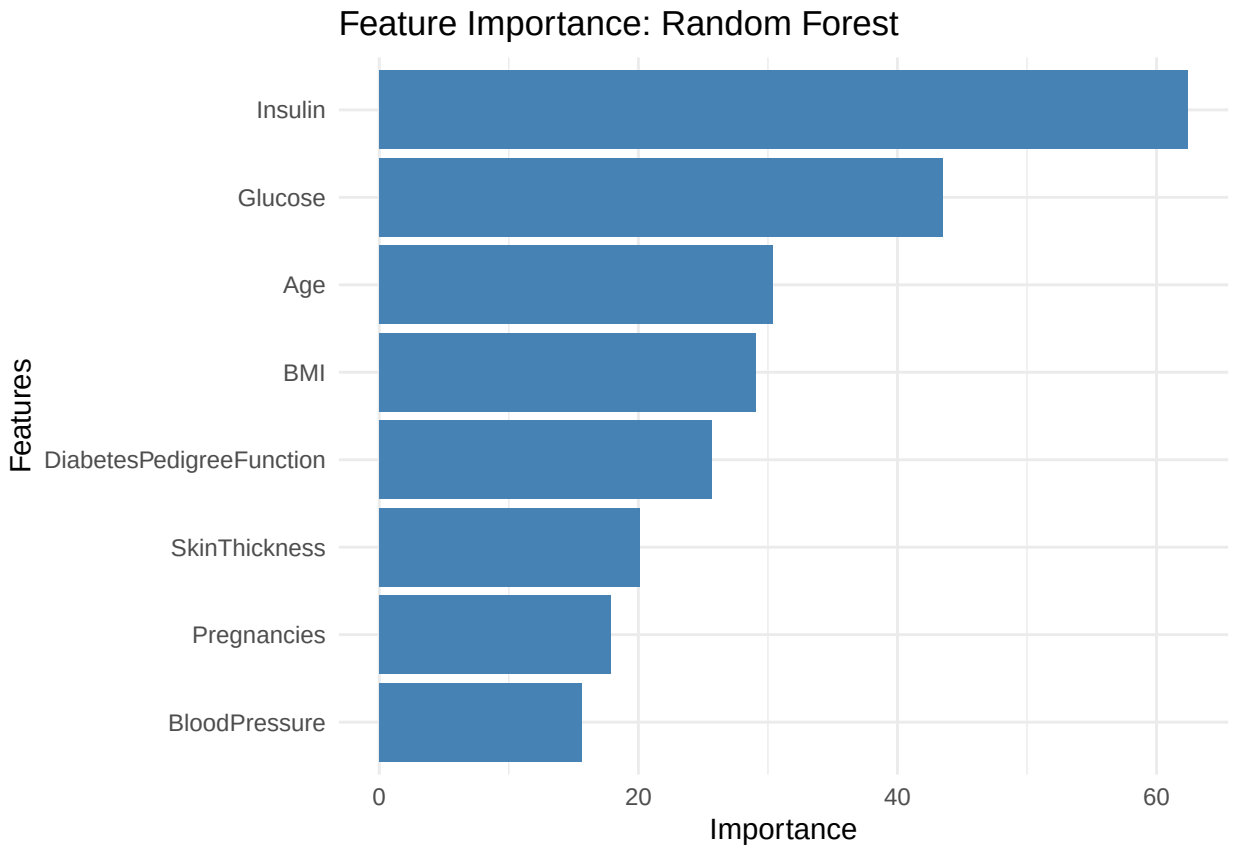
```
print(best_model)
```

```
##                    Model   Dataset  Accuracy  F1_Score Overall_Score
## Accuracy11 Random Forest Synthetic 0.9059829 0.9325153     0.9192491
```

```
# Bar chart for Overall Performance
ggplot(all_metrics, aes(x = Model, y = Overall_Score, fill = Dataset)) +
  geom_bar(stat = "identity", position = "dodge") +
  labs(title = "Overall Performance Comparison Across Models",
       x = "Model",
       y = "Overall Performance Score") +
  theme_minimal() +
  geom_text(aes(label = round(Overall_Score, 2)), position = position_dodge(width = 0.9), vjust = -0.5) +
  theme(axis.text.x = element_text(angle = 45, hjust = 1))
```




```r
# Function to calculate feature importance
calculate_feature_importance <- function(model, model_name, train_data) {
  importance_df <- data.frame(Feature = colnames(train_data[, -ncol(train_data)]), Importance = NA)

  if (model_name == "Random Forest") {
    rf_importance <- importance(model, type = 2)  # Mean Decrease Gini
    importance_df$Importance <- rf_importance[, "MeanDecreaseGini"]
  }

  return(importance_df)
}

# Function to plot feature importance
plot_feature_importance <- function(importance_df, model_name) {
  ggplot(importance_df, aes(x = reorder(Feature, Importance), y = Importance)) +
    geom_bar(stat = "identity", fill = "steelblue") +
    coord_flip() +
    labs(title = paste("Feature Importance:", model_name),
         x = "Features",
         y = "Importance") +
    theme_minimal()
}

# Random Forest Feature Importance
rf_model <- randomForest(as.factor(Outcome) ~ ., data = train_data_synthetic, importance = TRUE)
rf_importance <- calculate_feature_importance(rf_model, "Random Forest", train_data_synthetic)
plot_feature_importance(rf_importance, "Random Forest")
```

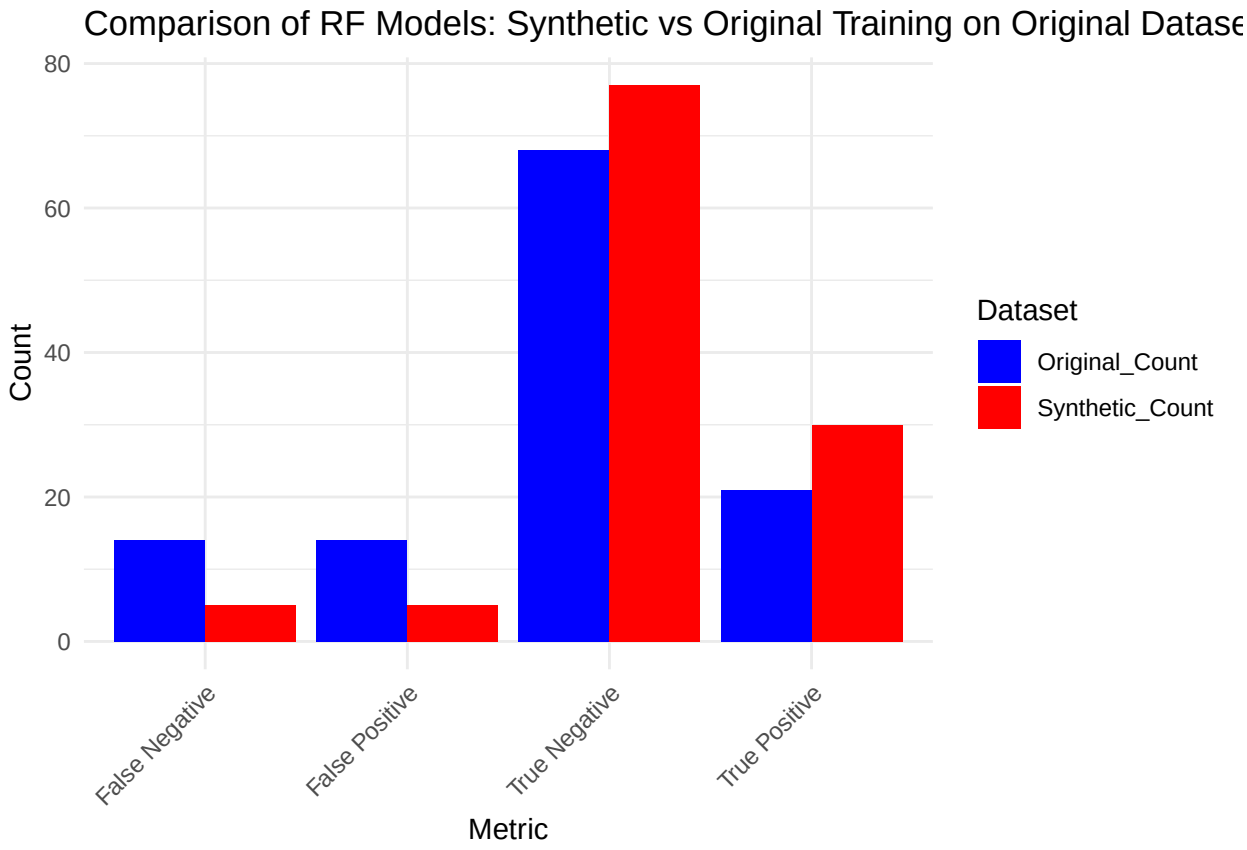



```r
# Step 8: Compare RF Models Trained on Synthetic vs. Original Datasets

# Train RF model on the synthetic dataset
rf_model_synthetic <- randomForest(as.factor(Outcome) ~ ., data = train_data_synthetic, ntree = 100)

# Train RF model on the original dataset
rf_model_original <- randomForest(as.factor(Outcome) ~ ., data = train_data_original, ntree = 100)

# Predict using the synthetic-trained model on the original dataset
synthetic_on_original_predictions <- predict(rf_model_synthetic, test_data_original)

# Predict using the original-trained model on the original dataset
original_on_original_predictions <- predict(rf_model_original, test_data_original)

# Compute confusion matrices
confusion_synthetic_on_original <- confusionMatrix(as.factor(synthetic_on_original_predictions), as.fac
confusion_original_on_original <- confusionMatrix(as.factor(original_on_original_predictions), as.facto

# Extract metrics from confusion matrices
extract_truth_table <- function(confusion) {
  matrix <- confusion$table
  TP <- matrix[2, 2]  # True Positives
  FP <- matrix[1, 2]  # False Positives
  TN <- matrix[1, 1]  # True Negatives
  FN <- matrix[2, 1]  # False Negatives

  data.frame(
```

```
    Metric = c("True Positive", "False Positive", "True Negative", "False Negative"),
    Count = c(TP, FP, TN, FN)
  )
}

truth_table_synthetic <- extract_truth_table(confusion_synthetic_on_original)
truth_table_original <- extract_truth_table(confusion_original_on_original)

# Combine truth tables for comparison
comparison_table <- truth_table_synthetic %>%
  rename(Synthetic_Count = Count) %>%
  inner_join(truth_table_original %>% rename(Original_Count = Count), by = "Metric")

# Print comparison table
print(comparison_table)
```

```
##           Metric Synthetic_Count Original_Count
## 1  True Positive              30             21
## 2 False Positive               5             14
## 3  True Negative              77             68
## 4 False Negative               5             14
```

```
# Visualize the comparison in a bar chart
comparison_long <- comparison_table %>%
  pivot_longer(cols = c(Synthetic_Count, Original_Count), names_to = "Dataset", values_to = "Count")

ggplot(comparison_long, aes(x = Metric, y = Count, fill = Dataset)) +
  geom_bar(stat = "identity", position = "dodge") +
  scale_fill_manual(values = c("Synthetic_Count" = "red", "Original_Count" = "blue")) +
  labs(title = "Comparison of RF Models: Synthetic vs Original Training on Original Dataset",
       x = "Metric",
       y = "Count") +
  theme_minimal() +
  theme(axis.text.x = element_text(angle = 45, hjust = 1))
```


Comparison of RF Models: Synthetic vs Original Training on Original Datase
Count
80
60
40
20
0
False Negative
False Positive
True Negative
True Positive
Metric
Dataset
Original_Count
Synthetic_Count