Mobile Device Usage and User Behavior Classification

Rohith Reddy

2024-11-07

Introduction:

In this project, we explore a Mobile Device Usage and User Behavior Dataset to predict user behavior classes using multiple machine learning models. We implement four different classification models: Random Forest, K-Nearest Neighbors (KNN), and Support Vector Machine (SVM) and Naive Bayes followed by performance evaluation and comparison.

Load necessary libraries

```
library(randomForest)

## randomForest 4.7-1.2

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

library(caret)

## Loading required package: ggplot2

##
## Attaching package: 'ggplot2'

## The following object is masked from 'package:randomForest':
##
##     margin

## Loading required package: lattice

library(class)
library(e1071)
library(ggplot2)
library(reshape2)
library(gmodels)
library(caTools)
```

1. Data Loading and Preprocessing

```
# Load the dataset
data <-
read.csv("C:/Users/Dell/Desktop/int234/project/user_behavior_dataset.csv")
colnames(data)

##  [1] "User.ID"                    "Device.Model"
##  [3] "Operating.System"           "App.Usage.Time..min.day."
```

```
##  [5] "Screen.On.Time..hours.day." "Battery.Drain..mAh.day."
##  [7] "Number.of.Apps.Installed"   "Data.Usage..MB.day."
##  [9] "Age"                        "Gender"
## [11] "User.Behavior.Class"
```

Loading Dataset of User Behavior to perform predictive analysis and see the result of different classifications models to find that which model is giving high accuracy.

```
# Ensure 'User.Behavior.Class' is a factor
data$User.Behavior.Class <- as.factor(data$User.Behavior.Class)
data <- data[sapply(data, is.atomic)]
#Key Features/Structure of the data
str(data)
```

```
## 'data.frame':    700 obs. of  11 variables:
##  $ User.ID                   : int  1 2 3 4 5 6 7 8 9 10 ...
##  $ Device.Model              : chr  "Google Pixel 5" "OnePlus 9" "Xiaomi
Mi 11" "Google Pixel 5" ...
##  $ Operating.System          : chr  "Android" "Android" "Android"
"Android" ...
##  $ App.Usage.Time..min.day.  : int  393 268 154 239 187 99 350 543 340 424
...
##  $ Screen.On.Time..hours.day.: num  6.4 4.7 4 4.8 4.3 2 7.3 11.4 7.7 6.6
...
##  $ Battery.Drain..mAh.day.   : int  1872 1331 761 1676 1367 940 1802 2956
2138 1957 ...
##  $ Number.of.Apps.Installed  : int  67 42 32 56 58 35 66 82 75 75 ...
##  $ Data.Usage..MB.day.       : int  1122 944 322 871 988 564 1054 1702
1053 1301 ...
##  $ Age                       : int  40 47 42 20 31 31 21 31 42 42 ...
##  $ Gender                    : chr  "Male" "Female" "Male" "Male" ...
##  $ User.Behavior.Class       : Factor w/ 5 levels "1","2","3","4",..: 4 3
2 3 3 2 4 5 4 4 ...
```

Split Dataset into Training and Testing Sets

```
set.seed(123)
train_indices <- createDataPartition(data$User.Behavior.Class, p = 0.7, list
= FALSE)
trainData <- data[train_indices, ]
testData <- data[-train_indices, ]
```

2. Model Training and Evaluation

2.1 Random Forest Model

```
rf_model <- randomForest(User.Behavior.Class ~ ., data = trainData, ntree =
500, mtry = 3, importance = TRUE)
print(rf_model)
```

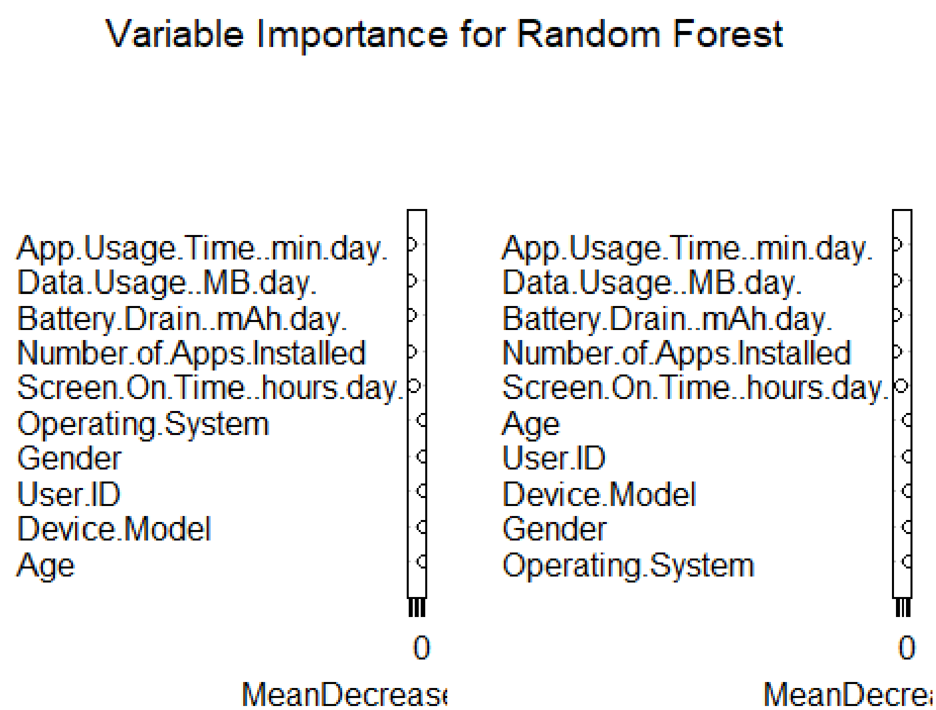
```
##
## Call:
##  randomForest(formula = User.Behavior.Class ~ ., data = trainData,
ntree = 500, mtry = 3, importance = TRUE)
##                Type of random forest: classification
##                      Number of trees: 500
## No. of variables tried at each split: 3
##
##         OOB estimate of  error rate: 0%
## Confusion matrix:
##    1   2   3  4  5 class.error
## 1 96   0   0  0  0           0
## 2  0 103   0  0  0           0
## 3  0   0 101  0  0           0
## 4  0   0   0 98  0           0
## 5  0   0   0  0 96           0
```

2.1.1 Analysis of Accuracy of KNN model

```r
# Predict and evaluate
predictions_rf <- predict(rf_model, newdata = testData)
conf_matrix_rf <- table(testData$User.Behavior.Class, predictions_rf)
accuracy_rf <- sum(diag(conf_matrix_rf)) / sum(conf_matrix_rf)
cat("Random Forest Accuracy:", accuracy_rf * 100, "%\n")

## Random Forest Accuracy: 100 %
```

2.1.2 Variable Importance Plot

```r
varImpPlot(rf_model, main = "Variable Importance for Random Forest")
```



2.2 K-Nearest Neighbors (KNN) Model

2.2.1 Preprocess: Normalize Features and Add Noise

```
normalize <- function(x) { (x - min(x)) / (max(x) - min(x)) }
numeric_columns <- sapply(data, is.numeric)
data_n <- as.data.frame(lapply(data[, numeric_columns], normalize))
data_n$User.Behavior.Class <- data$User.Behavior.Class
set.seed(123)
noise <- matrix(rnorm(n = nrow(data_n) * (ncol(data_n) - 1), mean = 0, sd =
0.1), nrow = nrow(data_n))
data_n[, -ncol(data_n)] <- data_n[, -ncol(data_n)] + noise
# Summary of data with noise
summary(data_n)
```

```
##     User.ID        App.Usage.Time..min.day. Screen.On.Time..hours.day.
##  Min.   :-0.1723   Min.   :-0.1888          Min.   :-0.1799
##  1st Qu.: 0.2444   1st Qu.: 0.1555          1st Qu.: 0.1496
##  Median : 0.5116   Median : 0.3679          Median : 0.3671
##  Mean   : 0.4997   Mean   : 0.4276          Mean   : 0.3923
##  3rd Qu.: 0.7452   3rd Qu.: 0.7186          3rd Qu.: 0.5852
##  Max.   : 1.2401   Max.   : 1.2058          Max.   : 1.1595
##  Battery.Drain..mAh.day. Number.of.Apps.Installed Data.Usage..MB.day.
##  Min.   :-0.1920         Min.   :-0.2230          Min.   :-0.1943
##  1st Qu.: 0.1803         1st Qu.: 0.1781          1st Qu.: 0.1138
##  Median : 0.4427         Median : 0.4438          Median : 0.2971
##  Mean   : 0.4513         Mean   : 0.4575          Mean   : 0.3441
```

```
##  3rd Qu.: 0.7172        3rd Qu.: 0.7089         3rd Qu.: 0.5407
##  Max.   : 1.1517        Max.   : 1.1525         Max.   : 1.0995
##       Age          User.Behavior.Class
##  Min.   :-0.2245   1:136
##  1st Qu.: 0.2444   2:146
##  Median : 0.4875   3:143
##  Mean   : 0.4980   4:139
##  3rd Qu.: 0.7485   5:136
##  Max.   : 1.1850
```

2.2.2 Train-Test Split and KNN Model

```
train_index <- sample(1:nrow(data_n), 0.7 * nrow(data_n))
data_train <- data_n[train_index, ]
data_test <- data_n[-train_index, ]
train_labels <- data_n$User.Behavior.Class[train_index]
test_labels <- data_n$User.Behavior.Class[-train_index]

k <- 50  # Adjust as needed
test_pred_knn <- knn(train = data_train[, -ncol(data_train)], test =
data_test[, -ncol(data_test)], cl = train_labels, k = k)
conf_matrix_knn <- table(test_labels, test_pred_knn)
conf_matrix_knn
```

```
##              test_pred_knn
## test_labels  1  2  3  4  5
##            1 40  2  0  0  0
##            2  4 38  1  0  0
##            3  0  2 42  0  0
##            4  0  0  3 40  0
##            5  0  0  0  1 38
```

2.2.3 Analysis of Accuracy of KNN model.

```
accuracy_knn <- sum(test_pred_knn == test_labels) / length(test_labels)
cat("KNN Accuracy:", round(accuracy_knn * 100, 2), "%\n")
```

```
## KNN Accuracy: 93.84 %
```

2.3 Naive Bayes Model with Feature Reduction

2.3.1 Features and Normalization

```
data_reduced <- data_n[, 1:2]  # Example: Adjust based on feature selection
split <- sample.split(data$User.Behavior.Class, SplitRatio = 0.7)
train_set <- subset(data_reduced, split == TRUE)
test_set <- subset(data_reduced, split == FALSE)
train_labels <- subset(data$User.Behavior.Class, split == TRUE)
test_labels <- subset(data$User.Behavior.Class, split == FALSE)
```

2.3.2 Train and Evaluate Naive Bayes Model

```r
nb_model <- naiveBayes(train_set, train_labels)
pred_labels_nb <- predict(nb_model, test_set)
conf_matrix_nb <- table(test_labels, pred_labels_nb)
conf_matrix_nb

##            pred_labels_nb
## test_labels  1  2  3  4  5
##           1 26 15  0  0  0
##           2 15 19 10  0  0
##           3  1  7 31  4  0
##           4  0  0  7 27  8
##           5  0  0  0 10 31
```

2.3.3 Analysis of Accuracy of Naive Bayes Model.

```r
accuracy_nb <- sum(diag(conf_matrix_nb)) / sum(conf_matrix_nb)
cat("Naive Bayes Accuracy with reduced features:", round(accuracy_nb * 100,
2), "%\n")

## Naive Bayes Accuracy with reduced features: 63.51 %
```

2.4 Support Vector Machine (SVM)

2.4.1 Train and Evaluate SVM Model

```r
svm_model <- svm(User.Behavior.Class ~ ., data = trainData, kernel =
"radial", cost = 0.09, scale = TRUE)
predictions_svm <- predict(svm_model, newdata = testData)
conf_matrix_svm <- table(testData$User.Behavior.Class, predictions_svm)
conf_matrix_svm

##    predictions_svm
##      1  2  3  4  5
##   1 39  1  0  0  0
##   2  0 43  0  0  0
##   3  0  0 42  0  0
##   4  0  0  0 41  0
##   5  0  0  0  0 40
```

2.4.2 Analysis of Accuracy of SVM

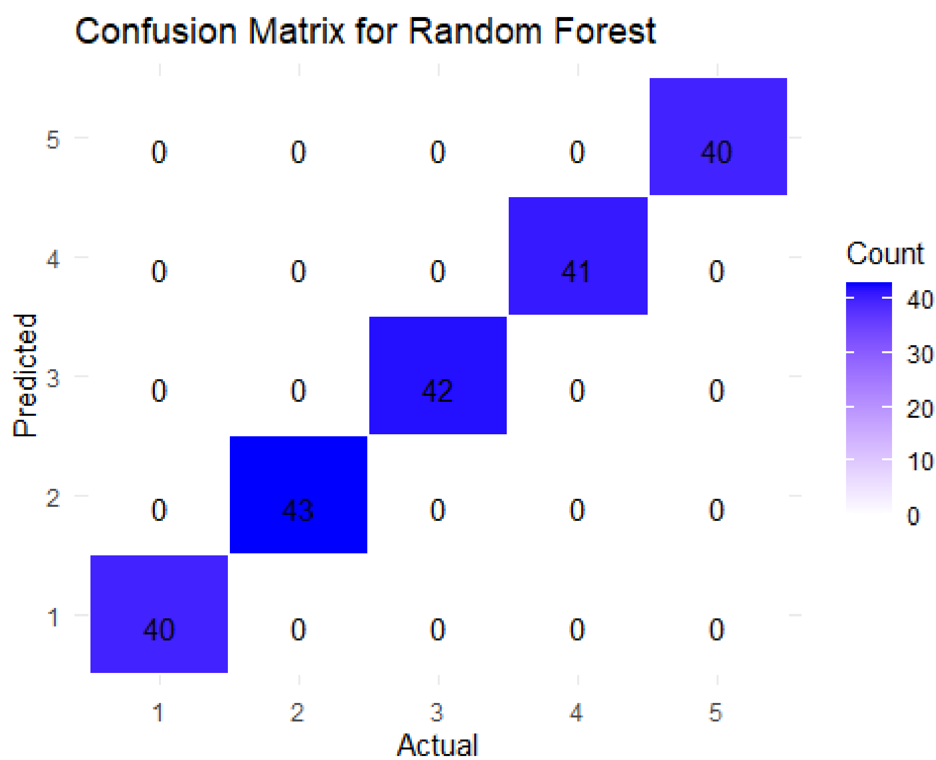
```r
accuracy_svm <- sum(diag(conf_matrix_svm)) / sum(conf_matrix_svm)
cat("SVM Accuracy:", accuracy_svm * 100, "%\n")

## SVM Accuracy: 99.51456 %
```

3.Visualizations of Confusion Matrices

3.1 Random Forest Confusion Matrix

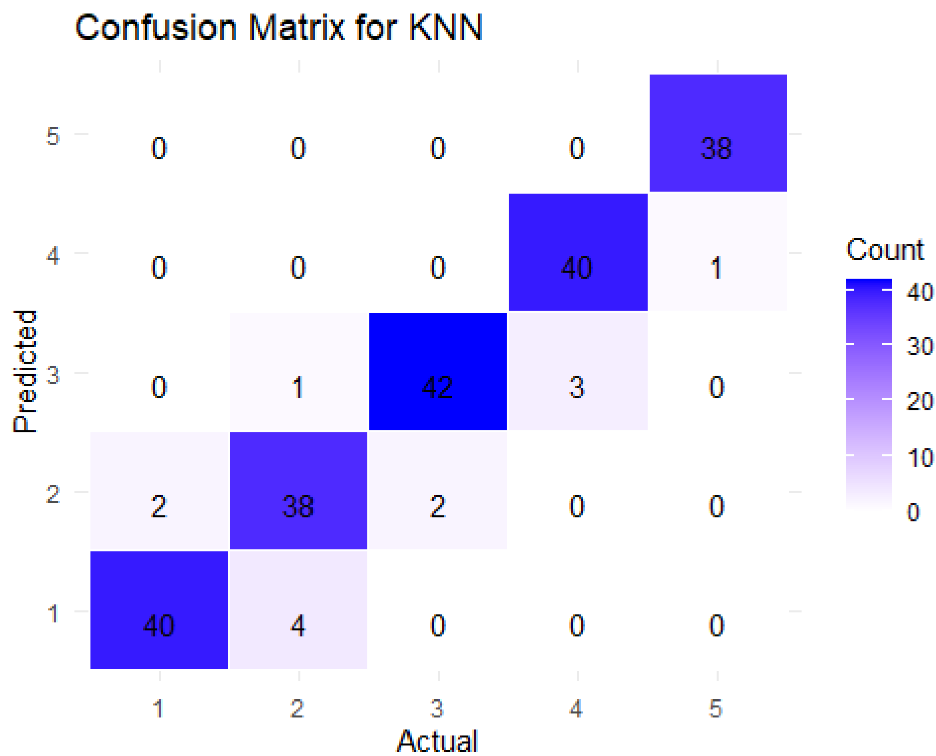
```r
conf_matrix_rf_df <- as.data.frame(conf_matrix_rf)
colnames(conf_matrix_rf_df) <- c("Actual", "Predicted", "Count")
```

```r
ggplot(data = conf_matrix_rf_df, aes(x = Actual, y = Predicted)) +
  geom_tile(aes(fill = Count), color = "white") +
  scale_fill_gradient(low = "white", high = "blue") +
  geom_text(aes(label = Count), vjust = 1) +
  labs(title = "Confusion Matrix for Random Forest", x = "Actual", y =
"Predicted") +
  theme_minimal()
```



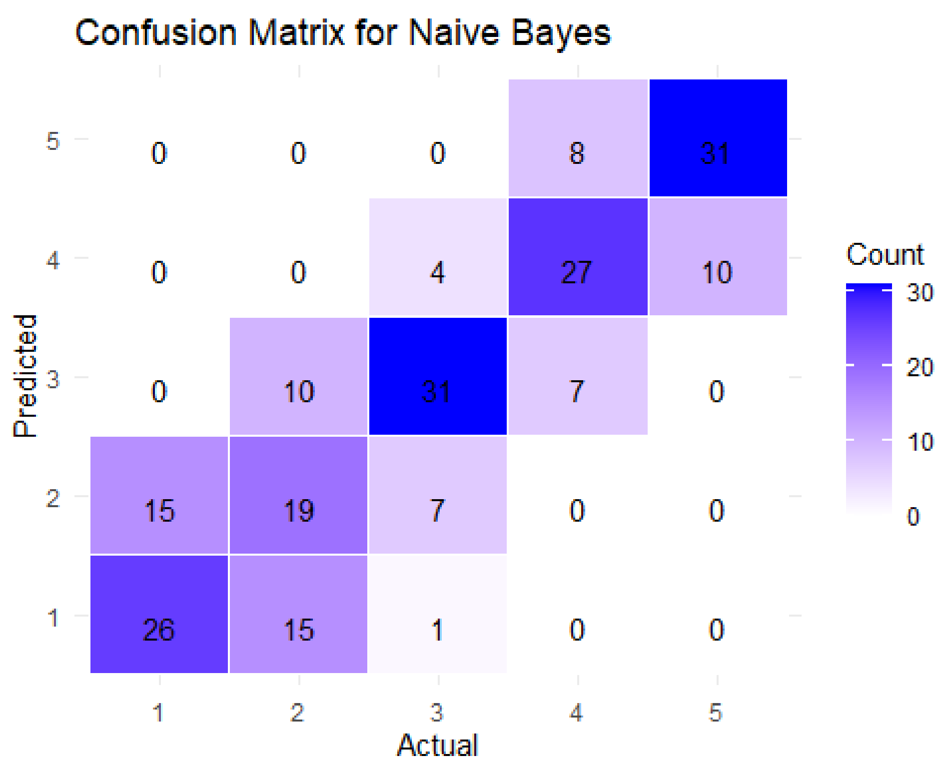
3.2 KNN Confusion Matrix

```r
conf_matrix_knn_df <- as.data.frame(conf_matrix_knn)
colnames(conf_matrix_knn_df) <- c("Actual", "Predicted", "Count")

ggplot(data = conf_matrix_knn_df, aes(x = Actual, y = Predicted)) +
  geom_tile(aes(fill = Count), color = "white") +
  scale_fill_gradient(low = "white", high = "blue") +
  geom_text(aes(label = Count), vjust = 1) +
  labs(title = "Confusion Matrix for KNN", x = "Actual", y = "Predicted") +
  theme_minimal()
```



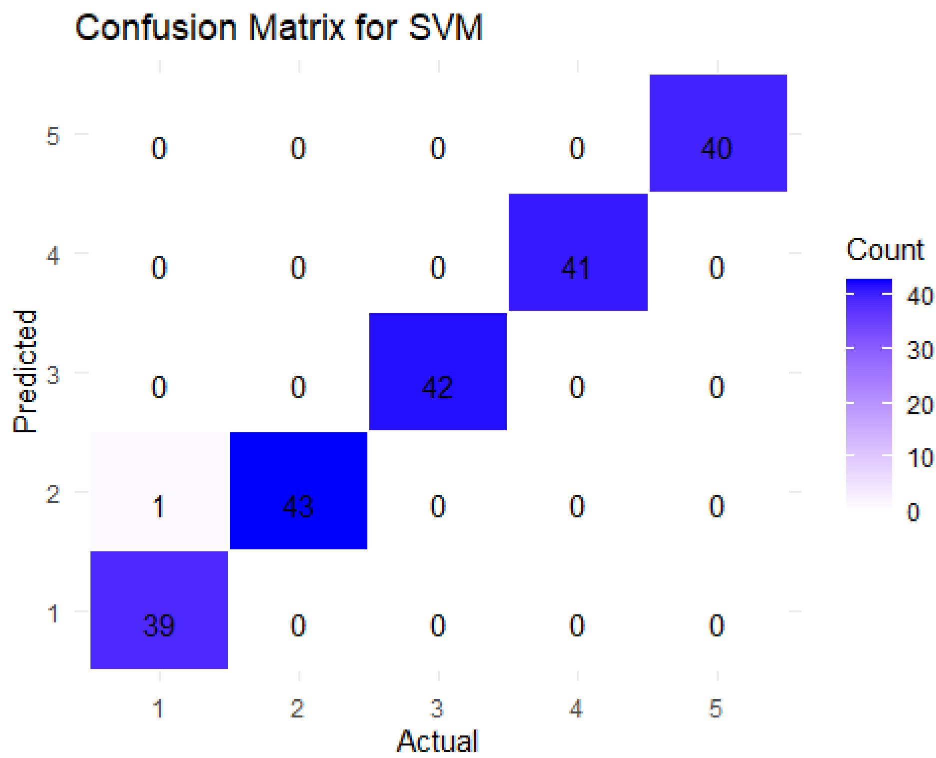
3.3 Naive Bayes Confusion Matrix

```r
conf_matrix_nb_df <- as.data.frame(conf_matrix_nb)
colnames(conf_matrix_nb_df) <- c("Actual", "Predicted", "Count")

ggplot(data = conf_matrix_nb_df, aes(x = Actual, y = Predicted)) +
  geom_tile(aes(fill = Count), color = "white") +
  scale_fill_gradient(low = "white", high = "blue") +
  geom_text(aes(label = Count), vjust = 1) +
  labs(title = "Confusion Matrix for Naive Bayes", x = "Actual", y =
"Predicted") +
  theme_minimal()
```



3.4 SVM Confusion Matrix

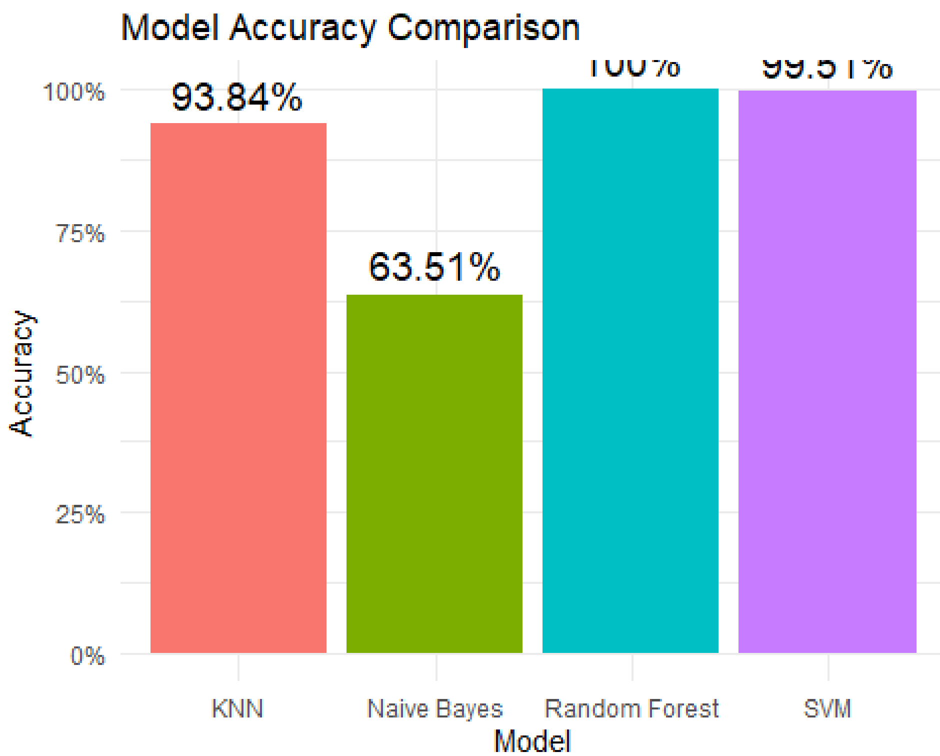
```
conf_matrix_svm_df <- as.data.frame(conf_matrix_svm)
colnames(conf_matrix_svm_df) <- c("Actual", "Predicted", "Count")

ggplot(data = conf_matrix_svm_df, aes(x = Actual, y = Predicted)) +
  geom_tile(aes(fill = Count), color = "white") +
  scale_fill_gradient(low = "white", high = "blue") +
  geom_text(aes(label = Count), vjust = 1) +
  labs(title = "Confusion Matrix for SVM", x = "Actual", y = "Predicted") +
  theme_minimal()
```



4. Model Comparison

4.1 Accuracy Comparison of Models

```
accuracies <- data.frame(
  Model = c("KNN", "Naive Bayes", "Random Forest", "SVM"),
  Accuracy = c(accuracy_knn, accuracy_nb, accuracy_rf, accuracy_svm)
)

ggplot(data = accuracies, aes(x = Model, y = Accuracy, fill = Model)) +
  geom_bar(stat = "identity") +
  geom_text(aes(label = paste0(round(Accuracy * 100, 2), "%")), vjust = -0.5,
size = 5) +
  scale_y_continuous(labels = scales::percent_format(accuracy = 1), limits =
c(0, 1)) +
  labs(title = "Model Accuracy Comparison", x = "Model", y = "Accuracy") +
  theme_minimal() +
  theme(legend.position = "none")
```


Model Accuracy Comparison
100%
75%
50%
25%
0%
Accuracy
93.84%
63.51%
KNN
Naive Bayes
Random Forest
SVM
Model