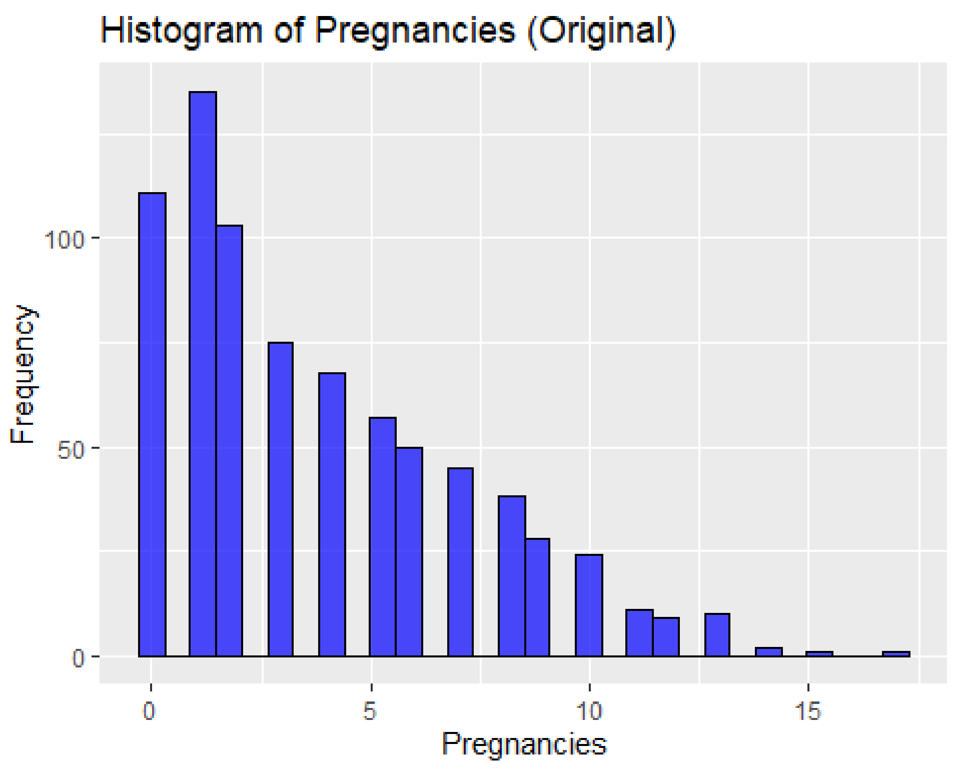
Poster Project

Leonel Salazar, Collin Real, Joaquin Ramirez, Seth Harris

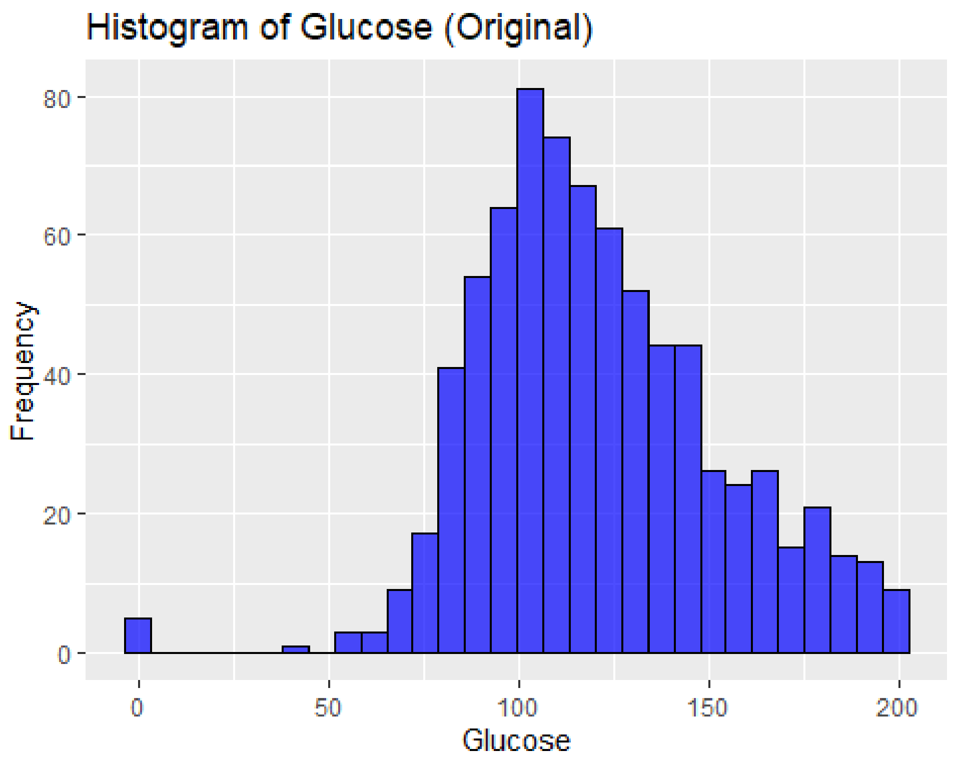
```
library(fmsb)

Warning: package 'fmsb' was built under R version 4.3.3

library(gbm)

Warning: package 'gbm' was built under R version 4.3.3

Loaded gbm 2.2.2

This version of gbm is no longer under development. Consider transitioning to
gbm3, https://github.com/gbm-developers/gbm3

library(here)

Warning: package 'here' was built under R version 4.3.3

here() starts at C:/Users/Leonel/Desktop/MSDA/MS Data Analytics/Current Class

library(ggplot2)

Warning: package 'ggplot2' was built under R version 4.3.3

library(gridExtra)
library(MASS)
library(corrplot)

Warning: package 'corrplot' was built under R version 4.3.3

corrplot 0.92 loaded

library(caret)

Warning: package 'caret' was built under R version 4.3.3

Loading required package: lattice

Warning: package 'lattice' was built under R version 4.3.3

Registered S3 methods overwritten by 'pROC':
  method    from
  print.roc fmsb
  plot.roc  fmsb

library(e1071)

Warning: package 'e1071' was built under R version 4.3.3
```

```
library(pROC)

Warning: package 'pROC' was built under R version 4.3.3

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


Attaching package: 'pROC'

The following object is masked from 'package:fmsb':

    roc

The following objects are masked from 'package:stats':

    cov, smooth, var

library(nnet)
library(pROC)
library(randomForest)

Warning: package 'randomForest' was built under R version 4.3.3

randomForest 4.7-1.1

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


Attaching package: 'randomForest'

The following object is masked from 'package:gridExtra':

    combine

The following object is masked from 'package:ggplot2':

    margin

library(class)

# Use 'here' to create the path to the dataset
diabetes <- here::here("diabetes.csv")

# Load the dataset and assign it to the variable 'diabetes'
diabetes <- read.csv(diabetes)

# View the first few rows of the dataset
head(diabetes)

  Pregnancies Glucose BloodPressure SkinThickness Insulin  BMI
1           6     148            72            35       0 33.6
2           1      85            66            29       0 26.6
```

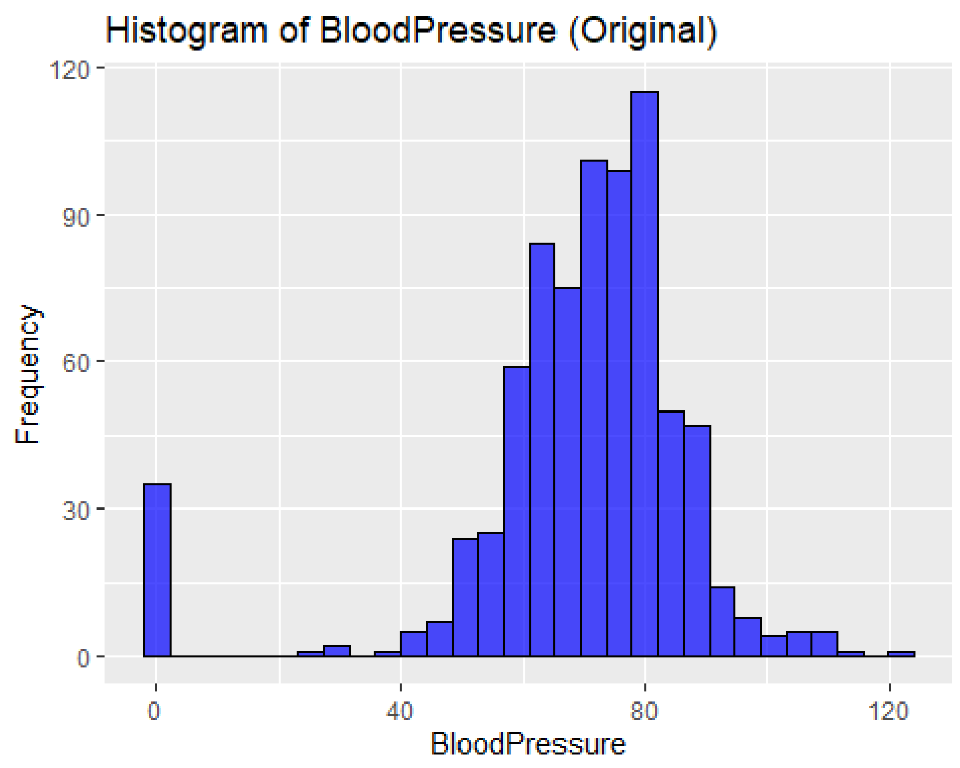
```
3           8     183          64           0       0 23.3
4           1      89          66          23      94 28.1
5           0     137          40          35     168 43.1
6           5     116          74           0       0 25.6
  DiabetesPedigreeFunction Age Outcome
1                    0.627  50       1
2                    0.351  31       0
3                    0.672  32       1
4                    0.167  21       0
5                    2.288  33       1
6                    0.201  30       0

# Convert the outcome variable to a factor
diabetes$Outcome <- as.factor(diabetes$Outcome)

numeric_columns <- names(diabetes)[names(diabetes) != "Outcome"]
diabetes[numeric_columns] <- lapply(diabetes[numeric_columns], as.numeric)

str(diabetes)

'data.frame':   768 obs. of  9 variables:
 $ Pregnancies             : num  6 1 8 1 0 5 3 10 2 8 ...
 $ Glucose                 : num  148 85 183 89 137 116 78 115 197 125 ...
 $ BloodPressure           : num  72 66 64 66 40 74 50 0 70 96 ...
 $ SkinThickness           : num  35 29 0 23 35 0 32 0 45 0 ...
 $ Insulin                 : num  0 0 0 94 168 0 88 0 543 0 ...
 $ BMI                     : num  33.6 26.6 23.3 28.1 43.1 25.6 31 35.3 30.5
0 ...
 $ DiabetesPedigreeFunction: num  0.627 0.351 0.672 0.167 2.288 ...
 $ Age                     : num  50 31 32 21 33 30 26 29 53 54 ...
 $ Outcome                 : Factor w/ 2 levels "0","1": 2 1 2 1 2 1 2 1 2 2
...

# Original data histograms
for (col in numeric_columns) {
  print(ggplot(diabetes, aes_string(x = col)) +
          geom_histogram(bins = 30, fill = "blue", color = "black", alpha =
0.7) +
          ggtitle(paste("Histogram of", col, "(Original)")) +
          xlab(col) +
          ylab("Frequency"))
}

Warning: `aes_string()` was deprecated in ggplot2 3.0.0.
ℹ Please use tidy evaluation idioms with `aes()`.
ℹ See also `vignette("ggplot2-in-packages")` for more information.
```

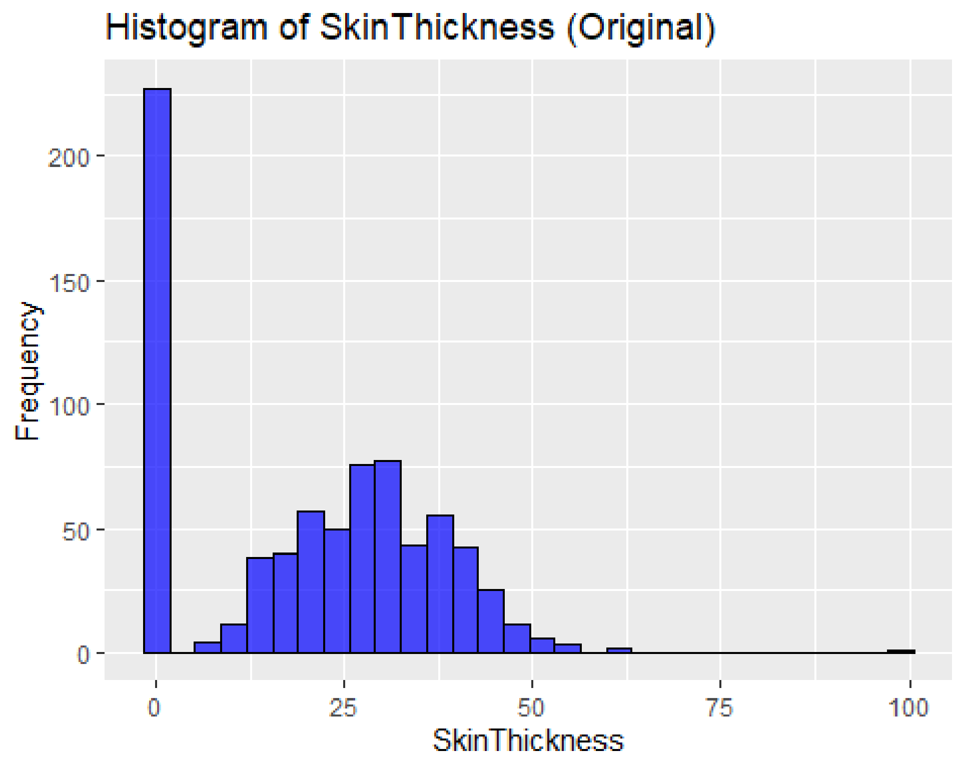


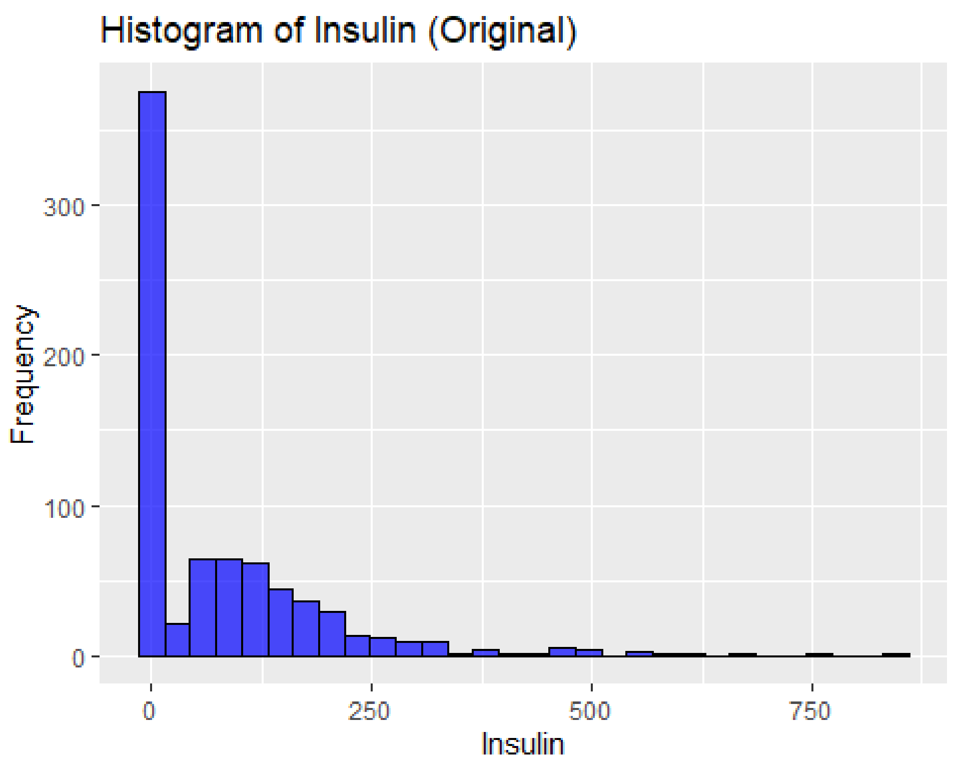
Histogram of Pregnancies (Original)
Frequency
0
50
100
0
5
10
15
Pregnancies



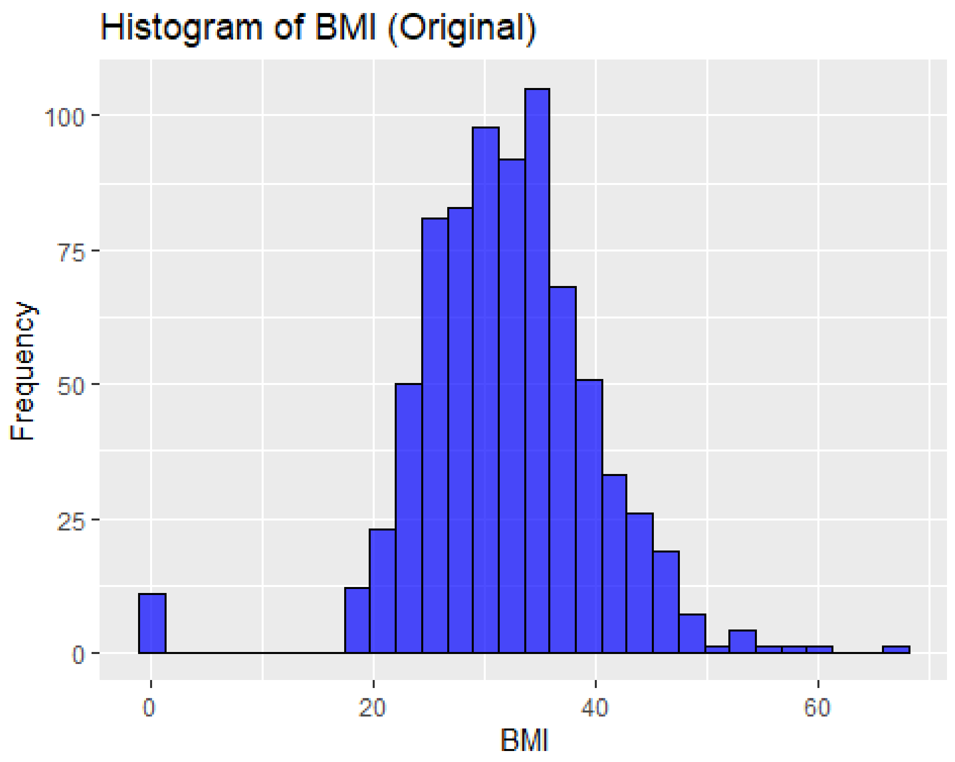
Histogram of Glucose (Original)
Frequency
0
20
40
60
80
0
50
100
150
200
Glucose


Histogram of BloodPressure (Original)
Frequency
120
90
60
30
0
0
40
80
120
BloodPressure

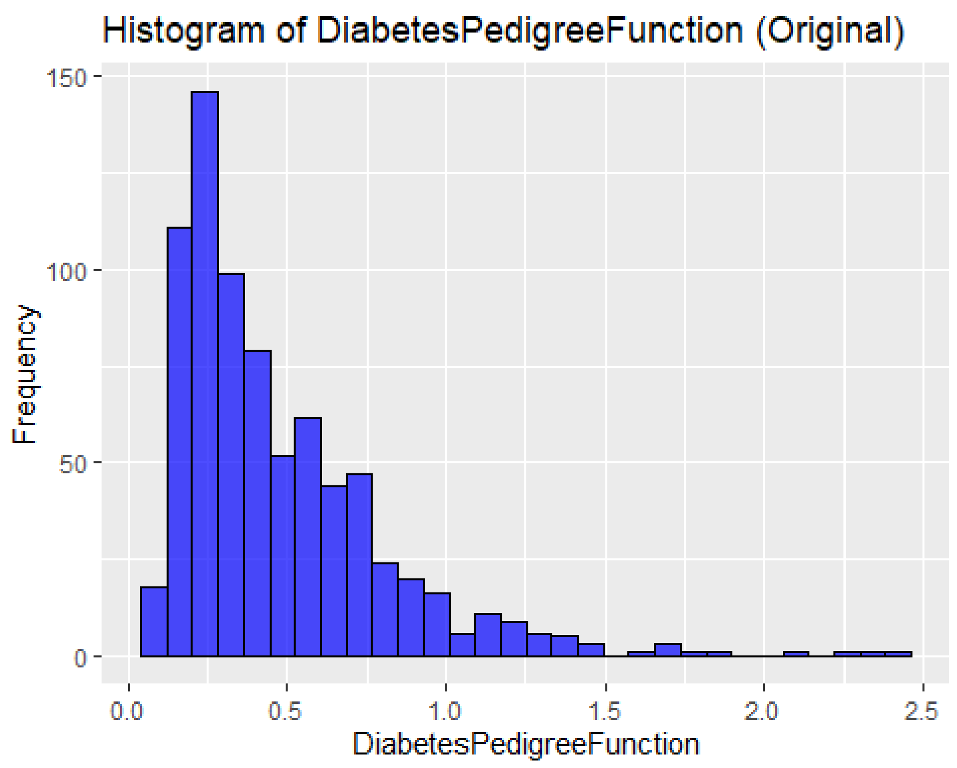

Histogram of SkinThickness (Original)
Frequency
200
150
100
50
0
0
25
50
75
100
SkinThickness

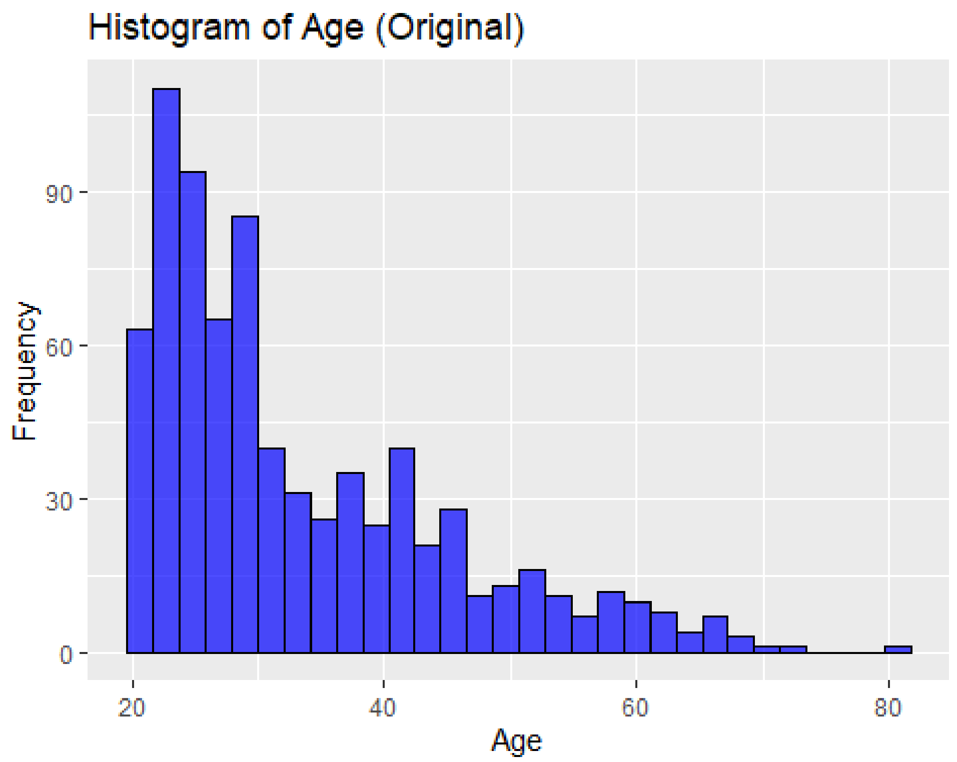


Histogram of Insulin (Original)
Frequency
300
200
100
0
0
250
500
750
Insulin

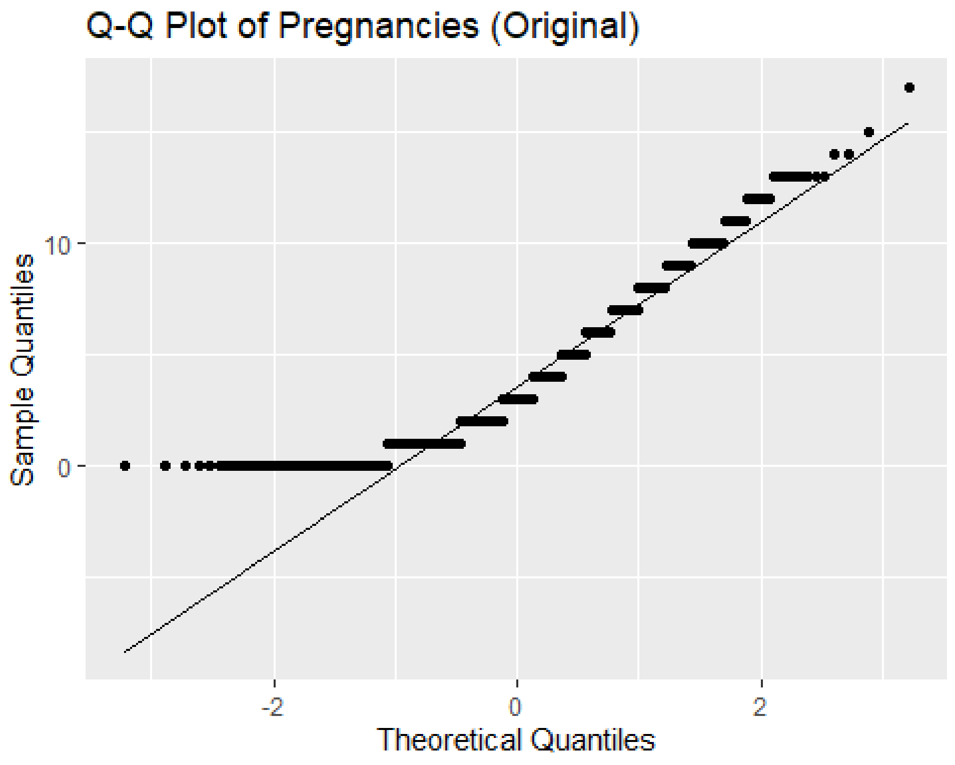


Histogram of BMI (Original)
Frequency
100
75
50
25
0
0
20
40
60
BMI

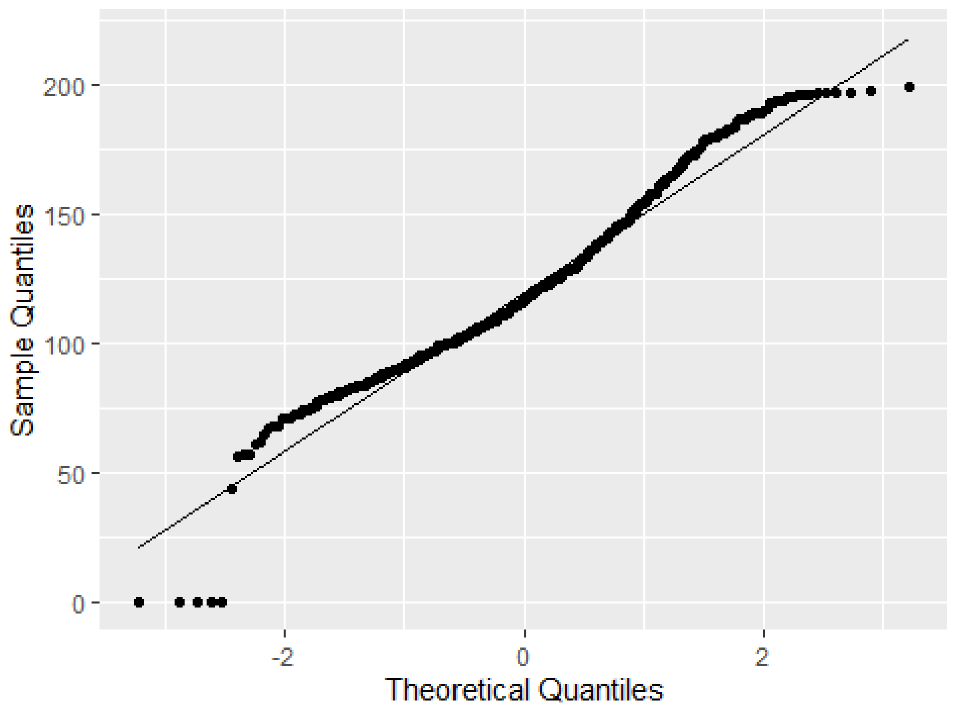




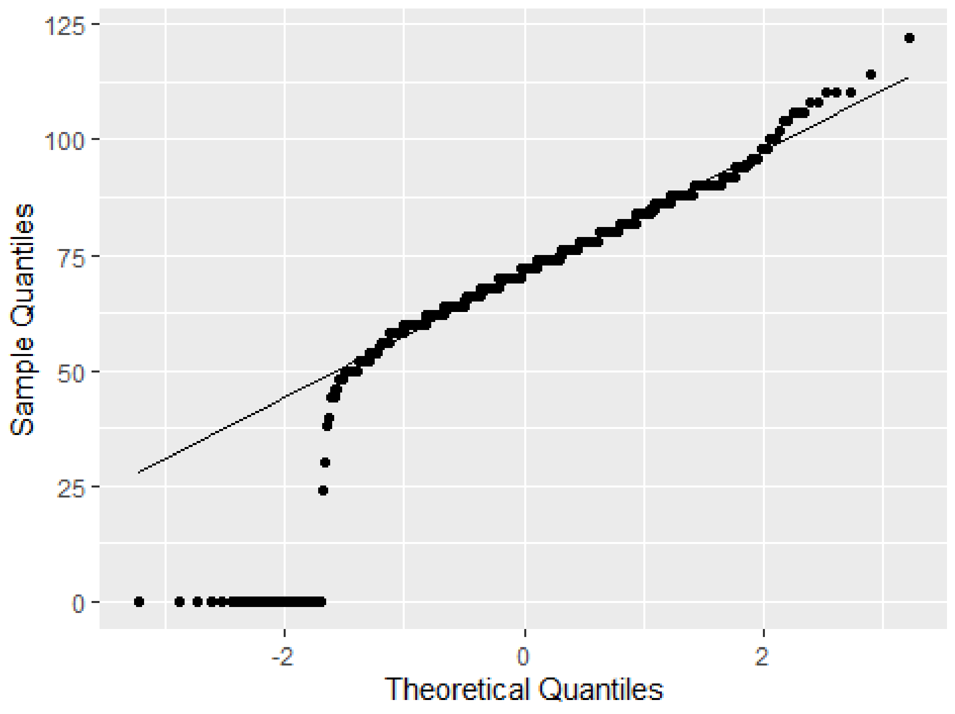
```
# Original data Q-Q plots
for (col in numeric_columns) {
  print(ggplot(diabetes, aes_string(sample = col)) +
          stat_qq() +
          stat_qq_line() +
          ggtitle(paste("Q-Q Plot of", col, "(Original)")) +
          xlab("Theoretical Quantiles") +
          ylab("Sample Quantiles"))
}
```



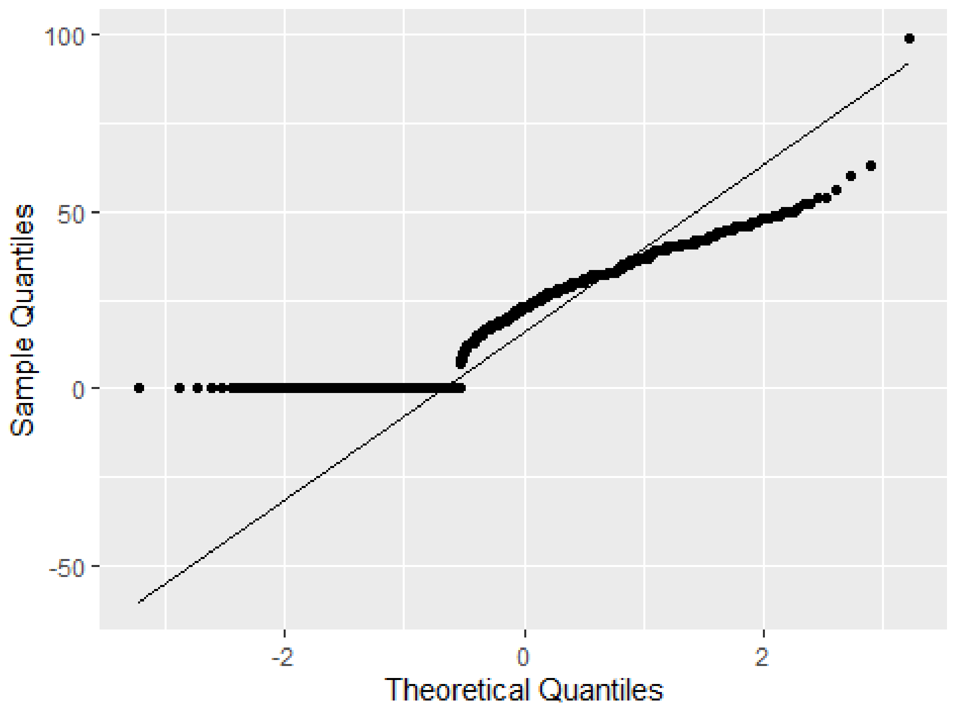
Q-Q Plot of Glucose (Original)



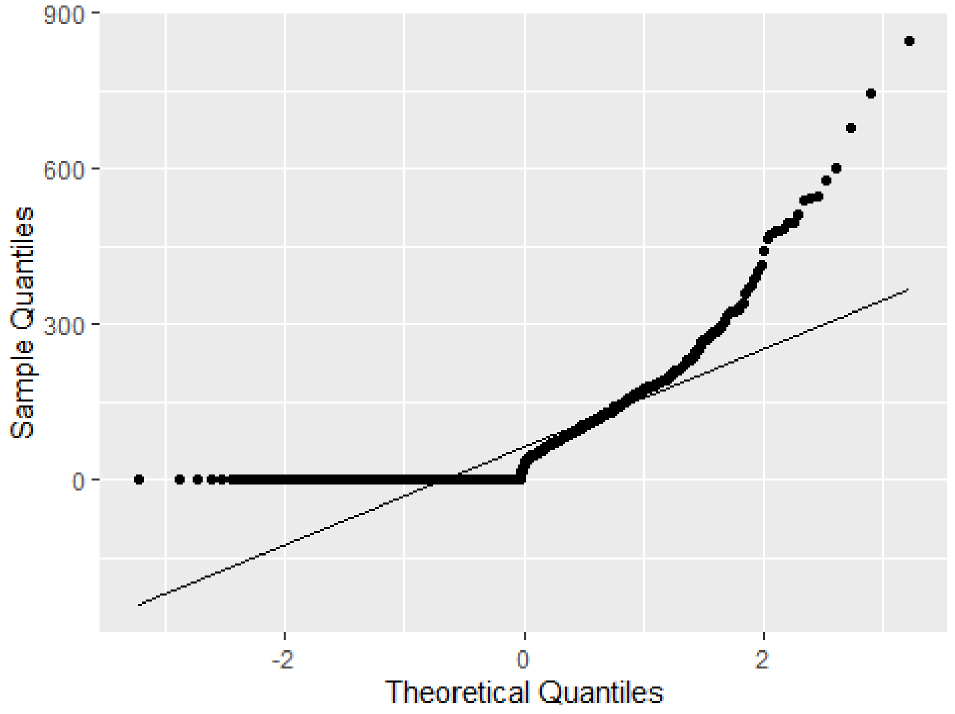
Q-Q Plot of BloodPressure (Original)

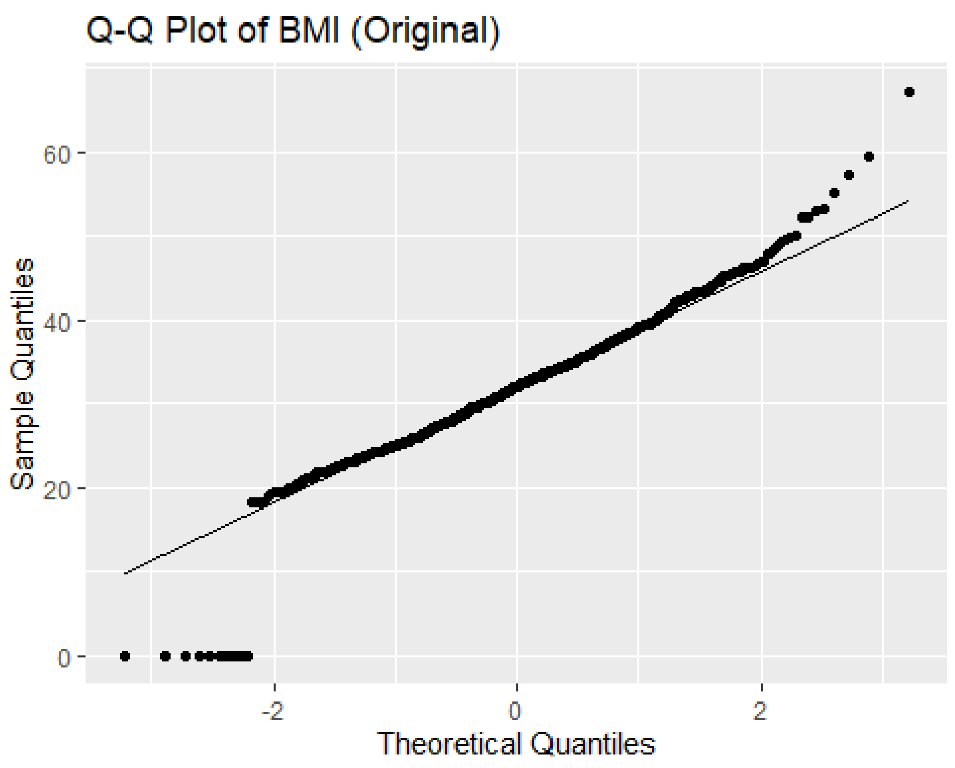


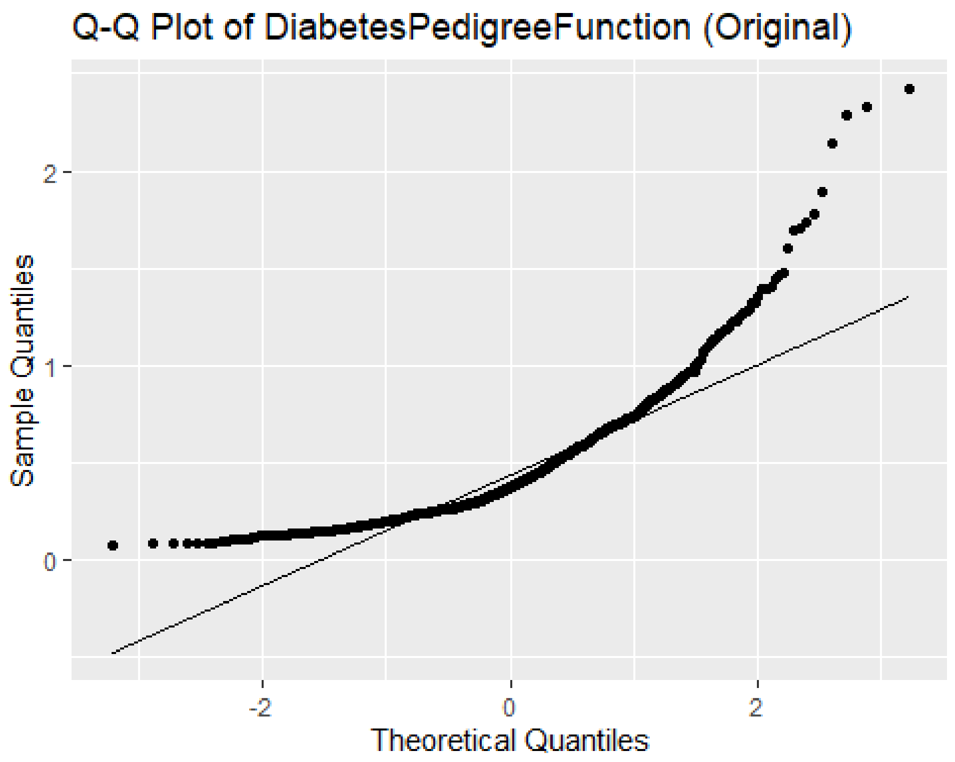
Q-Q Plot of SkinThickness (Original)

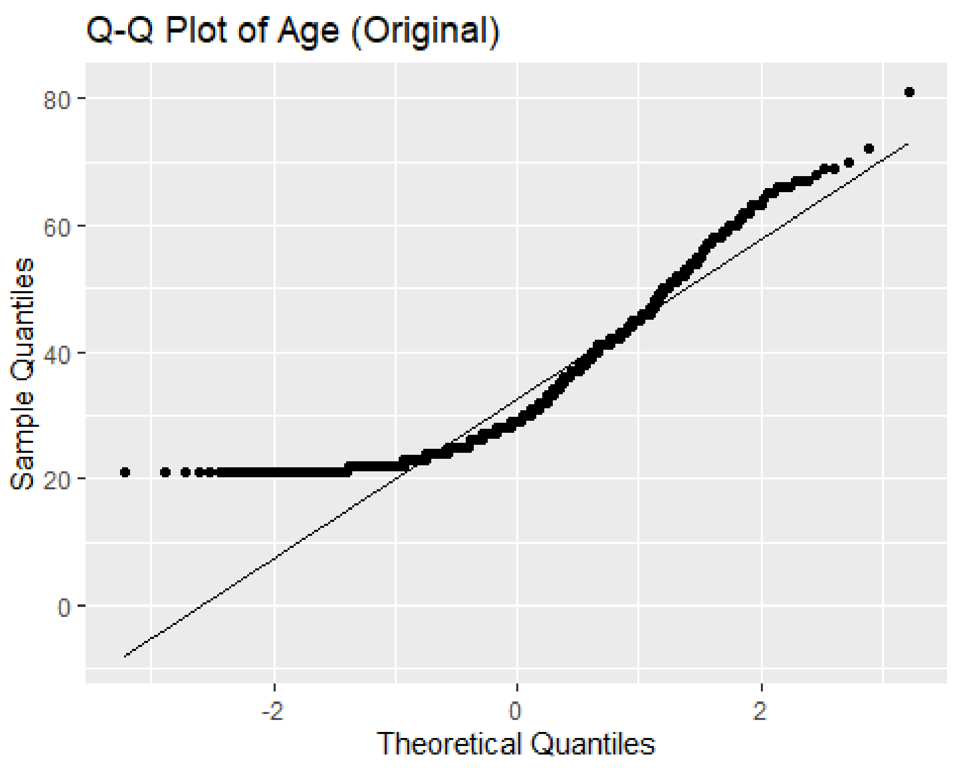


Q-Q Plot of Insulin (Original)




Q-Q Plot of BMI (Original)
60
40
20
0
Sample Quantiles
-2
0
2
Theoretical Quantiles


Q-Q Plot of DiabetesPedigreeFunction (Original)
2
1
0
Sample Quantiles
-2
0
2
Theoretical Quantiles



```r
# Load the dataset
diabetes_file_path <- here::here("diabetes.csv")
diabetes <- read.csv(diabetes_file_path)

# Convert the Outcome variable to a factor
diabetes$Outcome <- as.factor(diabetes$Outcome)

# Set seed for reproducibility
set.seed(123)

# Create training (20%) and testing (80%) indices
train_index <- createDataPartition(diabetes$Outcome, p = 0.2, list = FALSE)

# Split the data
train_data <- diabetes[train_index, ]
test_data <- diabetes[-train_index, ]

# Verify the split
cat("Training set size:", nrow(train_data), "\n")
```

```
Training set size: 154
```

```r
cat("Testing set size:", nrow(test_data), "\n")
```

```
Testing set size: 614
```

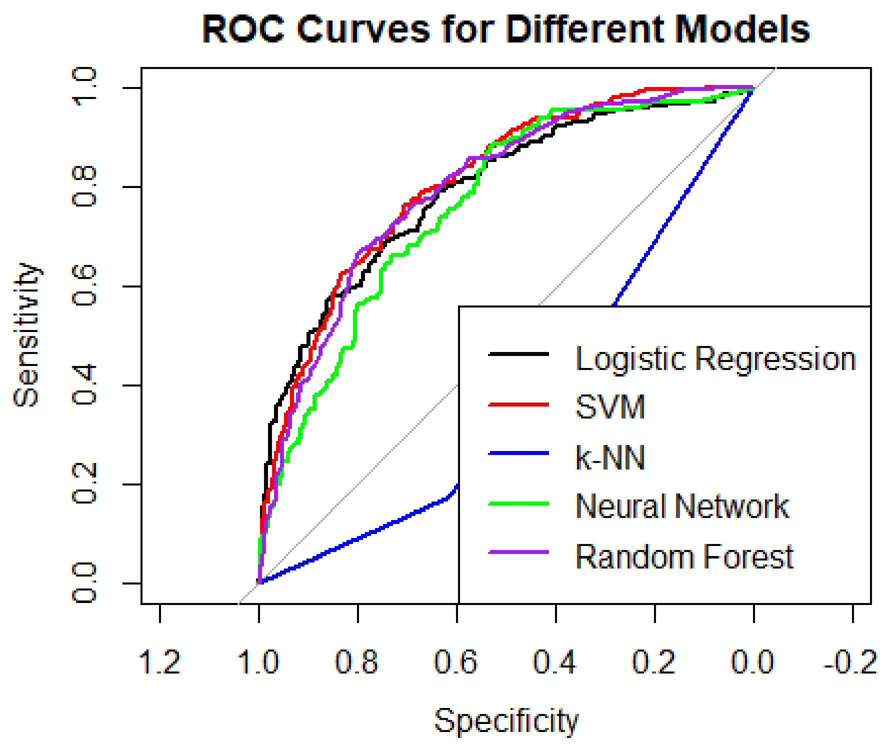
```r
# Normalize the predictor variables for neural network
preproc <- preProcess(train_data[, -which(names(train_data) == "Outcome")],
method = c("center", "scale"))
train_data_norm <- predict(preproc, train_data[, -which(names(train_data) ==
"Outcome")])
test_data_norm <- predict(preproc, test_data[, -which(names(test_data) ==
"Outcome")])

# Add the Outcome variable back to the normalized data
train_data_norm$Outcome <- train_data$Outcome
test_data_norm$Outcome <- test_data$Outcome

# Fit logistic regression model
lr_model <- glm(Outcome ~ ., data = train_data, family = binomial)
lr_prob_predictions <- predict(lr_model, newdata = test_data, type =
"response")

# Evaluate logistic regression model
lr_predictions <- ifelse(lr_prob_predictions > 0.5, 1, 0)
lr_confusion <- confusionMatrix(as.factor(lr_predictions), test_data$Outcome)
lr_accuracy <- lr_confusion$overall["Accuracy"]
lr_precision <- lr_confusion$byClass["Pos Pred Value"]
lr_recall <- lr_confusion$byClass["Sensitivity"]
lr_f1 <- 2 * (lr_precision * lr_recall) / (lr_precision + lr_recall)

# Logistic Regression ROC Curve
lr_roc_curve <- roc(test_data$Outcome, lr_prob_predictions)

Setting levels: control = 0, case = 1

Setting direction: controls < cases

# Fit SVM model
svm_model <- svm(Outcome ~ ., data = train_data, kernel = "radial",
probability = TRUE)
svm_predictions <- predict(svm_model, newdata = test_data, probability =
TRUE)
svm_prob_predictions <- attr(svm_predictions, "probabilities")[, 2]

# Evaluate SVM model
svm_confusion <- confusionMatrix(as.factor(ifelse(svm_prob_predictions > 0.5,
1, 0)), test_data$Outcome)
svm_accuracy <- svm_confusion$overall["Accuracy"]
svm_precision <- svm_confusion$byClass["Pos Pred Value"]
svm_recall <- svm_confusion$byClass["Sensitivity"]
svm_f1 <- 2 * (svm_precision * svm_recall) / (svm_precision + svm_recall)

# SVM ROC Curve
svm_roc_curve <- roc(test_data$Outcome, svm_prob_predictions)
```

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

Setting direction: controls < cases

# Fit k-NN model (k = 5)
knn_model <- knn(train = train_data[, -which(names(train_data) ==
"Outcome")],
                 test = test_data[, -which(names(test_data) == "Outcome")],
                 cl = train_data$Outcome, k = 5, prob = TRUE)
knn_prob_predictions <- attr(knn_model, "prob")

# Evaluate k-NN model
knn_confusion <- confusionMatrix(knn_model, test_data$Outcome)
knn_accuracy <- knn_confusion$overall["Accuracy"]
knn_precision <- knn_confusion$byClass["Pos Pred Value"]
knn_recall <- knn_confusion$byClass["Sensitivity"]
knn_f1 <- 2 * (knn_precision * knn_recall) / (knn_precision + knn_recall)

# k-NN ROC Curve
knn_roc_curve <- roc(test_data$Outcome, knn_prob_predictions)

Setting levels: control = 0, case = 1

Setting direction: controls < cases

# Fit neural network model
nn_model <- nnet(Outcome ~ ., data = train_data_norm, size = 5, maxit = 200,
decay = 0.01, linout = FALSE)

# weights:  51
initial  value 97.168442
iter  10 value 52.844198
iter  20 value 35.472041
iter  30 value 26.606381
iter  40 value 22.522330
iter  50 value 21.913196
iter  60 value 21.767813
iter  70 value 21.758312
iter  80 value 21.753819
iter  90 value 21.753647
final  value 21.753645
converged

nn_prob_predictions <- predict(nn_model, newdata = test_data_norm, type =
"raw")

# Ensure predictions are factors with the same levels as the actual outcomes
nn_predictions <- ifelse(nn_prob_predictions > 0.5, 1, 0)
nn_predictions <- as.factor(nn_predictions)
levels(nn_predictions) <- levels(test_data_norm$Outcome)
```

```r
# Evaluate neural network model
nn_confusion <- confusionMatrix(nn_predictions, test_data_norm$Outcome)
nn_accuracy <- nn_confusion$overall["Accuracy"]
nn_precision <- nn_confusion$byClass["Pos Pred Value"]
nn_recall <- nn_confusion$byClass["Sensitivity"]
nn_f1 <- 2 * (nn_precision * nn_recall) / (nn_precision + nn_recall)

# Neural Network ROC Curve
nn_roc_curve <- roc(test_data_norm$Outcome, as.numeric(nn_prob_predictions))

Setting levels: control = 0, case = 1

Setting direction: controls < cases

# Fit random forest model
rf_model <- randomForest(Outcome ~ ., data = train_data, ntree = 100, mtry =
3, importance = TRUE)
rf_prob_predictions <- predict(rf_model, newdata = test_data, type =
"prob")[, 2]

# Evaluate random forest model
rf_confusion <- confusionMatrix(predict(rf_model, newdata = test_data),
test_data$Outcome)
rf_accuracy <- rf_confusion$overall["Accuracy"]
rf_precision <- rf_confusion$byClass["Pos Pred Value"]
rf_recall <- rf_confusion$byClass["Sensitivity"]
rf_f1 <- 2 * (rf_precision * rf_recall) / (rf_precision + rf_recall)

# Random Forest ROC Curve
rf_roc_curve <- roc(test_data$Outcome, rf_prob_predictions)

Setting levels: control = 0, case = 1

Setting direction: controls < cases

# Create a data frame to compare models
model_comparison1 <- data.frame(
  Model = c("Logistic Regression", "SVM", "k-NN", "Neural Network", "Random
Forest"),
  Accuracy = c(lr_accuracy, svm_accuracy, knn_accuracy, nn_accuracy,
rf_accuracy),
  Precision = c(lr_precision, svm_precision, knn_precision, nn_precision,
rf_precision),
  Recall = c(lr_recall, svm_recall, knn_recall, nn_recall, rf_recall),
  F1_Score = c(lr_f1, svm_f1, knn_f1, nn_f1, rf_f1)
)

# Print the comparison table
print(model_comparison1)
```

```
                Model  Accuracy Precision Recall  F1_Score
1 Logistic Regression 0.7524430 0.7743363 0.8750 0.8215962
2                 SVM 0.7459283 0.7735426 0.8625 0.8156028
3                k-NN 0.6954397 0.7371938 0.8275 0.7797409
4      Neural Network 0.7035831 0.7725000 0.7725 0.7725000
5       Random Forest 0.7345277 0.7749420 0.8350 0.8038508
```

```
# Plot ROC Curves
plot(lr_roc_curve, col = "black", main = "ROC Curves for Different Models")
plot(svm_roc_curve, add = TRUE, col = "red")
plot(knn_roc_curve, add = TRUE, col = "blue")
plot(nn_roc_curve, add = TRUE, col = "green")
plot(rf_roc_curve, add = TRUE, col = "purple")
legend("bottomright", legend = c("Logistic Regression", "SVM", "k-NN",
"Neural Network", "Random Forest"),
       col = c("black", "red", "blue", "green", "purple"), lwd = 2)
```



```
# Print AUC values
cat("AUC for Logistic Regression:", auc(lr_roc_curve), "\n")

AUC for Logistic Regression: 0.7883411

cat("AUC for SVM:", auc(svm_roc_curve), "\n")

AUC for SVM: 0.8054673

cat("AUC for k-NN:", auc(knn_roc_curve), "\n")
```

```
AUC for k-NN: 0.3767407

cat("AUC for Neural Network:", auc(nn_roc_curve), "\n")

AUC for Neural Network: 0.7585864

cat("AUC for Random Forest:", auc(rf_roc_curve), "\n")

AUC for Random Forest: 0.7944042
```

Removed all Zeroes

```
library(here)
library(caret)
library(e1071)
library(class)
library(nnet)
library(randomForest)
library(pROC)
library(ggplot2)
library(gridExtra)

# Load the dataset
diabetes_file_path <- here::here("diabetes.csv")
diabetes <- read.csv(diabetes_file_path)

# Convert the Outcome variable to a factor
diabetes$Outcome <- as.factor(diabetes$Outcome)

# Identify the columns to exclude from zero-checking
exclude_columns <- c("Outcome")

# Check how many rows have a 0 in any column except the Outcome column
rows_with_zero <- apply(diabetes[, !names(diabetes) %in% exclude_columns], 1,
function(row) any(row == 0))

# Remove rows with a 0 in any column except the Outcome column
diabetes_cleaned <- diabetes[!rows_with_zero, ]

# Display the number of remaining observations
cat("Number of observations after removing rows with 0 in any column
(excluding Outcome):", nrow(diabetes_cleaned), "\n")

Number of observations after removing rows with 0 in any column (excluding
Outcome): 336

# Set seed for reproducibility
set.seed(123)

# Create training (80%) and testing (20%) indices
train_index <- createDataPartition(diabetes_cleaned$Outcome, p = 0.8, list =
FALSE)
```

```
# Split the data
train_data <- diabetes_cleaned[train_index, ]
test_data <- diabetes_cleaned[-train_index, ]

# Verify the split
cat("Training set size:", nrow(train_data), "\n")
```

```
Training set size: 269
```

```
cat("Testing set size:", nrow(test_data), "\n")
```

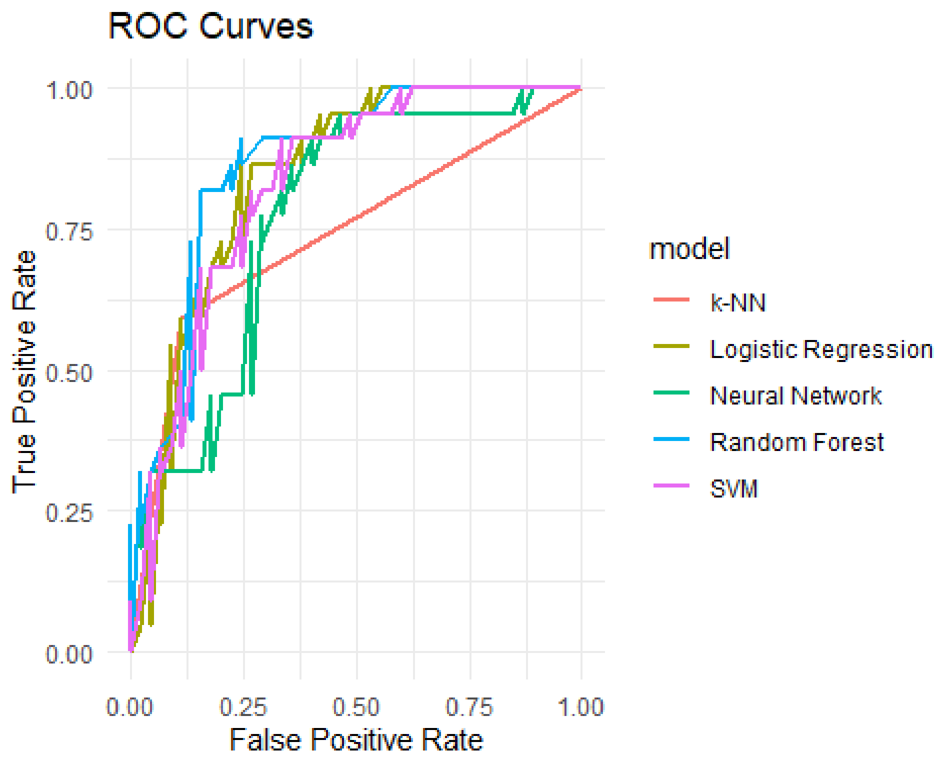
```
Testing set size: 67
```

```
# Normalize the predictor variables for neural network and k-NN
preproc <- preProcess(train_data[, -which(names(train_data) == "Outcome")],
method = c("center", "scale"))
train_data_norm <- predict(preproc, train_data[, -which(names(train_data) ==
"Outcome")])
test_data_norm <- predict(preproc, test_data[, -which(names(test_data) ==
"Outcome")])
diabetes_norm <- predict(preproc, diabetes_cleaned[, -
which(names(diabetes_cleaned) == "Outcome")])

# Add the Outcome variable back to the normalized data
train_data_norm$Outcome <- train_data$Outcome
test_data_norm$Outcome <- test_data$Outcome
diabetes_norm$Outcome <- diabetes_cleaned$Outcome

# Fit logistic regression model
lr_model <- glm(Outcome ~ ., data = train_data, family = binomial)
lr_prob_predictions <- predict(lr_model, newdata = test_data, type =
"response")
lr_predictions <- as.factor(ifelse(lr_prob_predictions > 0.5, 1, 0))

# Fit SVM model
svm_model <- svm(Outcome ~ ., data = train_data, kernel = "radial",
probability = TRUE)
svm_prob_predictions <- predict(svm_model, newdata = test_data, probability =
TRUE)
svm_prob_predictions <- attr(svm_prob_predictions, "probabilities")[, 2]
svm_predictions <- as.factor(ifelse(svm_prob_predictions > 0.5, 1, 0))

# Fit k-NN model (k = 5)
knn_model <- knn(train = train_data_norm[, -which(names(train_data_norm) ==
"Outcome")],
                 test = test_data_norm[, -which(names(test_data_norm) ==
"Outcome")],
                 cl = train_data_norm$Outcome, k = 5)
knn_predictions <- knn_model
```

```
# Fit neural network model
nn_model <- nnet(Outcome ~ ., data = train_data_norm, size = 5, maxit = 200,
decay = 0.01, linout = FALSE)

# weights:  51
initial  value 163.827891
iter  10 value 106.844055
iter  20 value 88.132080
iter  30 value 82.934518
iter  40 value 80.964255
iter  50 value 77.907908
iter  60 value 74.862650
iter  70 value 73.097675
iter  80 value 72.205309
iter  90 value 71.664927
iter 100 value 71.391457
iter 110 value 71.354651
iter 120 value 71.342521
iter 130 value 71.338160
iter 140 value 71.337916
final  value 71.337895
converged

nn_prob_predictions <- predict(nn_model, newdata = test_data_norm, type =
"raw")
nn_predictions <- as.factor(ifelse(nn_prob_predictions > 0.5, 1, 0))

# Fit random forest model
rf_model <- randomForest(Outcome ~ ., data = train_data, ntree = 100, mtry =
3, importance = TRUE)
rf_prob_predictions <- predict(rf_model, newdata = test_data, type =
"prob")[, 2]
rf_predictions <- as.factor(ifelse(rf_prob_predictions > 0.5, 1, 0))

# Evaluate Models
evaluate_model <- function(predictions, prob_predictions, true_labels) {
  confusion <- confusionMatrix(predictions, true_labels)
  accuracy <- confusion$overall["Accuracy"]
  precision <- confusion$byClass["Pos Pred Value"]
  recall <- confusion$byClass["Sensitivity"]
  f1 <- 2 * (precision * recall) / (precision + recall)
  auc_val <- auc(roc(true_labels, prob_predictions))
  list(confusion = confusion, accuracy = accuracy, precision = precision,
recall = recall, f1 = f1, auc = auc_val)
}

lr_eval <- evaluate_model(lr_predictions, lr_prob_predictions,
test_data$Outcome)
```

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

Setting direction: controls < cases

svm_eval <- evaluate_model(svm_predictions, svm_prob_predictions,
test_data$Outcome)

Setting levels: control = 0, case = 1
Setting direction: controls < cases

knn_eval <- evaluate_model(knn_predictions, as.numeric(knn_predictions),
test_data$Outcome)

Setting levels: control = 0, case = 1
Setting direction: controls < cases

nn_eval <- evaluate_model(nn_predictions, as.numeric(nn_prob_predictions),
test_data$Outcome)

Setting levels: control = 0, case = 1
Setting direction: controls < cases

rf_eval <- evaluate_model(rf_predictions, rf_prob_predictions,
test_data$Outcome)

Setting levels: control = 0, case = 1
Setting direction: controls < cases

# Create a data frame to compare models
model_comparison <- data.frame(
  Model = c("Logistic Regression", "SVM", "k-NN", "Neural Network", "Random
Forest"),
  Accuracy = c(lr_eval$accuracy, svm_eval$accuracy, knn_eval$accuracy,
nn_eval$accuracy, rf_eval$accuracy),
  Precision = c(lr_eval$precision, svm_eval$precision, knn_eval$precision,
nn_eval$precision, rf_eval$precision),
  Recall = c(lr_eval$recall, svm_eval$recall, knn_eval$recall,
nn_eval$recall, rf_eval$recall),
  F1_Score = c(lr_eval$f1, svm_eval$f1, knn_eval$f1, nn_eval$f1, rf_eval$f1),
  AUC = c(lr_eval$auc, svm_eval$auc, knn_eval$auc, nn_eval$auc, rf_eval$auc)
)

# Print the comparison table
print(model_comparison)

                Model  Accuracy Precision    Recall  F1_Score       AUC
1 Logistic Regression 0.7910448 0.8039216 0.9111111 0.8541667 0.8434343
2                 SVM 0.7611940 0.7843137 0.8888889 0.8333333 0.8292929
3                k-NN 0.7910448 0.8163265 0.8888889 0.8510638 0.7398990
4      Neural Network 0.6567164 0.7391304 0.7555556 0.7472527 0.7767677
5       Random Forest 0.7910448 0.8297872 0.8666667 0.8478261 0.8671717
```

```r
# Extract ROC data
extract_roc_data <- function(roc_object) {
  data.frame(
    specificity = roc_object$specificities,
    sensitivity = roc_object$sensitivities,
    model = deparse(substitute(roc_object))
  )
}

lr_roc_data <- extract_roc_data(roc(test_data$Outcome, lr_prob_predictions))

Setting levels: control = 0, case = 1
Setting direction: controls < cases

svm_roc_data <- extract_roc_data(roc(test_data$Outcome,
svm_prob_predictions))

Setting levels: control = 0, case = 1
Setting direction: controls < cases

knn_roc_data <- extract_roc_data(roc(test_data$Outcome,
as.numeric(knn_predictions)))

Setting levels: control = 0, case = 1
Setting direction: controls < cases

nn_roc_data <- extract_roc_data(roc(test_data$Outcome,
as.numeric(nn_prob_predictions)))

Setting levels: control = 0, case = 1
Setting direction: controls < cases

rf_roc_data <- extract_roc_data(roc(test_data$Outcome, rf_prob_predictions))

Setting levels: control = 0, case = 1
Setting direction: controls < cases

# Combine ROC data
roc_data <- rbind(
  transform(lr_roc_data, model = "Logistic Regression"),
  transform(svm_roc_data, model = "SVM"),
  transform(knn_roc_data, model = "k-NN"),
  transform(nn_roc_data, model = "Neural Network"),
  transform(rf_roc_data, model = "Random Forest")
)

# Plot ROC Curves
roc_plot <- ggplot(roc_data, aes(x = 1 - specificity, y = sensitivity, color
= model)) +
  geom_line(size = 1) +
  labs(x = "False Positive Rate", y = "True Positive Rate", title = "ROC
Curves") +
  theme_minimal()

Warning: Using `size` aesthetic for lines was deprecated in ggplot2 3.4.0.
ℹ Please use `linewidth` instead.

# Display the ROC plot
print(roc_plot)
```



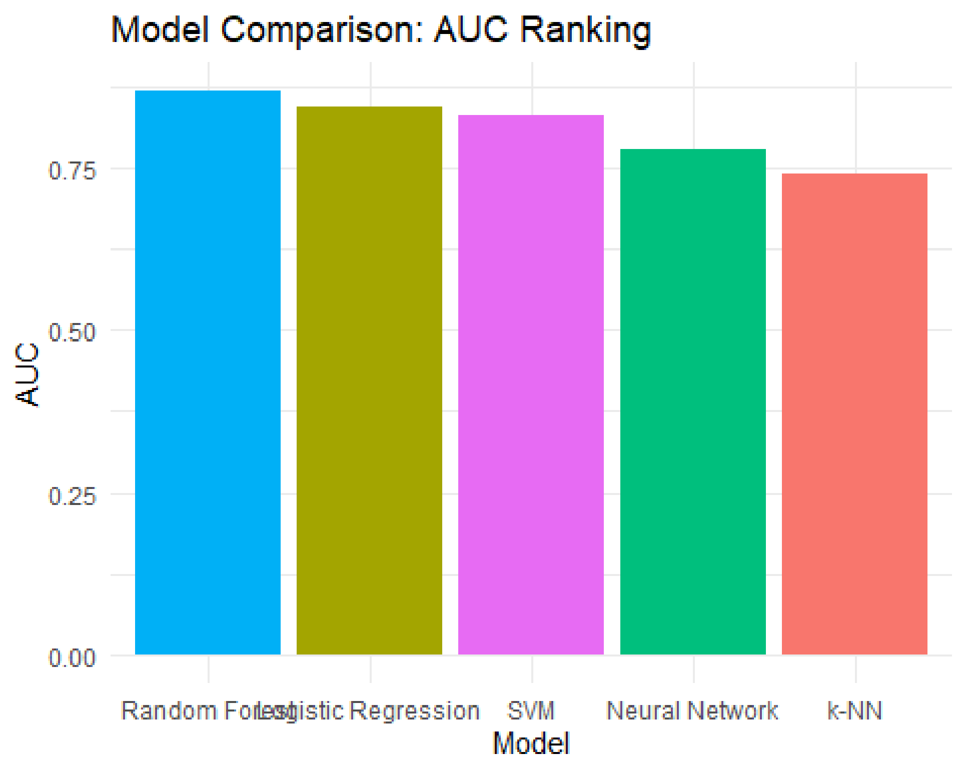
```
# Generate predictions for the entire dataset
lr_prob_predictions <- predict(lr_model, newdata = diabetes_cleaned, type =
"response")
lr_predictions <- as.factor(ifelse(lr_prob_predictions > 0.5, 1, 0))

svm_prob_predictions <- predict(svm_model, newdata = diabetes_cleaned,
probability = TRUE)
svm_prob_predictions <- attr(svm_prob_predictions, "probabilities")[, 2]
svm_predictions <- as.factor(ifelse(svm_prob_predictions > 0.5, 1, 0))

knn_predictions <- knn(train = train_data_norm[, -
which(names(train_data_norm) == "Outcome")],
                       test = diabetes_norm[, -which(names(diabetes_norm) ==
"Outcome")],
                       cl = train_data_norm$Outcome, k = 5)

nn_prob_predictions <- predict(nn_model, newdata = diabetes_norm, type =
"raw")
```

```r
nn_predictions <- as.factor(ifelse(nn_prob_predictions > 0.5, 1, 0))

rf_prob_predictions <- predict(rf_model, newdata = diabetes_cleaned, type =
"prob")[, 2]
rf_predictions <- as.factor(ifelse(rf_prob_predictions > 0.5, 1, 0))

# Create a new dataset with original outcomes and predictions
diabetes_pred <- diabetes_cleaned
diabetes_pred$LR_Prediction <- lr_predictions
diabetes_pred$SVM_Prediction <- svm_predictions
diabetes_pred$KNN_Prediction <- knn_predictions
diabetes_pred$NN_Prediction <- nn_predictions
diabetes_pred$RF_Prediction <- rf_predictions

# Display the first few rows of the new dataset
head(diabetes_pred)
```

```
   Pregnancies Glucose BloodPressure SkinThickness Insulin  BMI
4            1      89            66            23      94 28.1
7            3      78            50            32      88 31.0
9            2     197            70            45     543 30.5
14           1     189            60            23     846 30.1
15           5     166            72            19     175 25.8
19           1     103            30            38      83 43.3
   DiabetesPedigreeFunction Age Outcome LR_Prediction SVM_Prediction
4                     0.167  21       0             0              0
7                     0.248  26       1             0              0
9                     0.158  53       1             1              1
14                    0.398  59       1             1              1
15                    0.587  51       1             1              1
19                    0.183  33       0             0              0
   KNN_Prediction NN_Prediction RF_Prediction
4               0             0             0
7               0             0             1
9               1             1             1
14              1             1             1
15              1             1             1
19              0             0             0
```

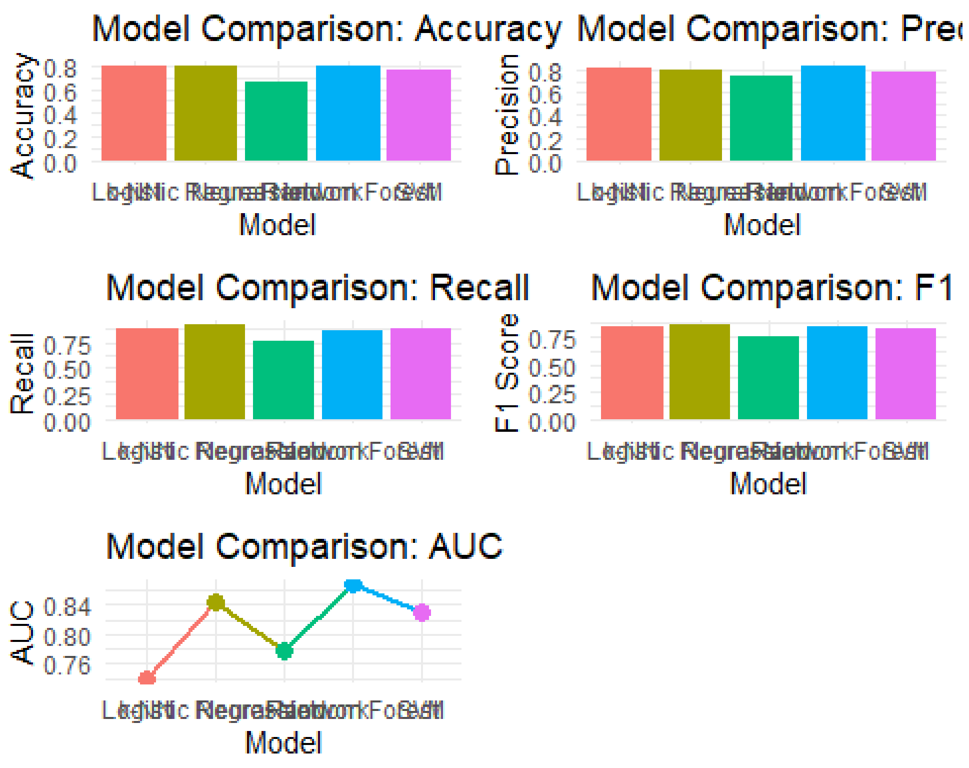
```r
library(ggplot2)

# Create a copy of the model_comparison dataframe
model_comparison_ordered <- model_comparison

# Add a column for the overall ranking based on AUC
model_comparison_ordered$Rank <- rank(-model_comparison_ordered$AUC)

# Sort the dataframe by the ranking
model_comparison_ordered <-
model_comparison_ordered[order(model_comparison_ordered$Rank), ]
```

```
# Plot the rankings
ggplot(model_comparison_ordered, aes(x = reorder(Model, -AUC), y = AUC, fill
= Model)) +
  geom_bar(stat = "identity") +
  labs(title = "Model Comparison: AUC Ranking", x = "Model", y = "AUC") +
  theme_minimal() +
  theme(legend.position = "none")
```

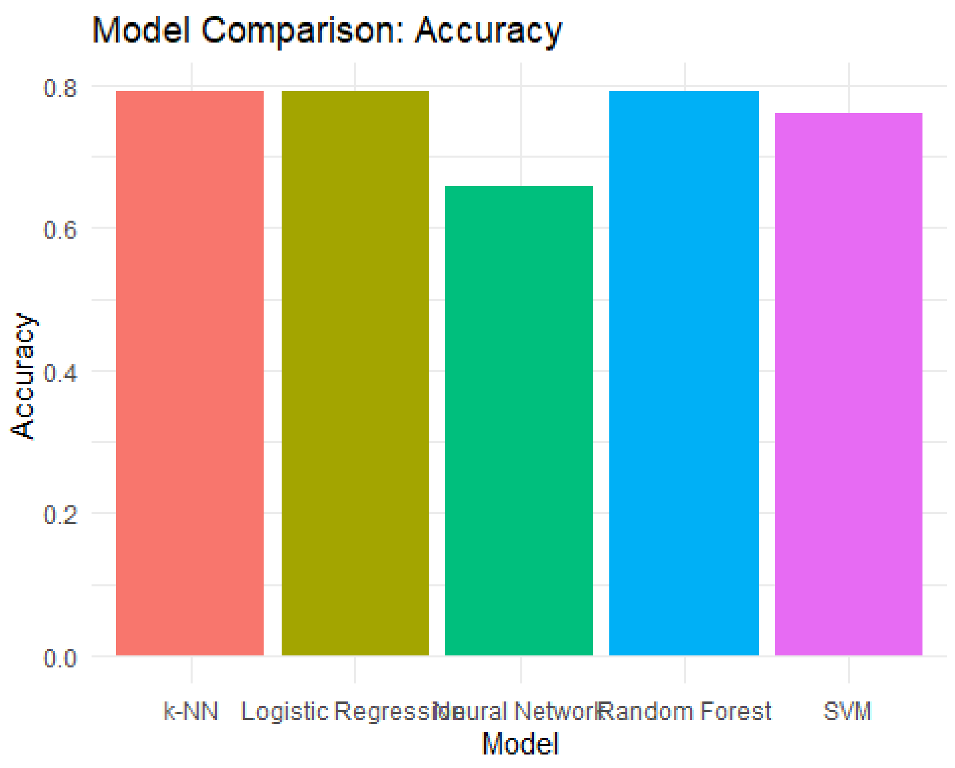


```
library(gridExtra)

# Create individual plots
plot1 <- ggplot(model_comparison, aes(x = Model, y = Accuracy, fill = Model))
+
  geom_bar(stat = "identity") +
  labs(title = "Model Comparison: Accuracy", x = "Model", y = "Accuracy") +
  theme_minimal() +
  theme(legend.position = "none")

plot2 <- ggplot(model_comparison, aes(x = Model, y = Precision, fill =
Model)) +
  geom_bar(stat = "identity") +
  labs(title = "Model Comparison: Precision", x = "Model", y = "Precision") +
  theme_minimal() +
  theme(legend.position = "none")
```

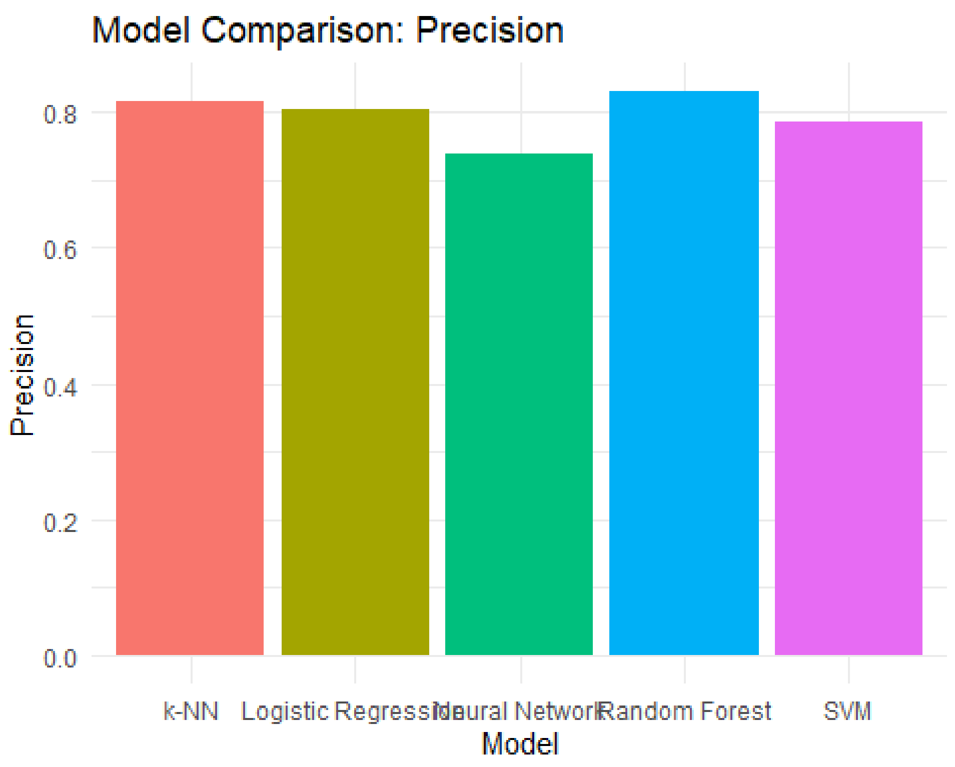
```r
plot3 <- ggplot(model_comparison, aes(x = Model, y = Recall, fill = Model)) +
  geom_bar(stat = "identity") +
  labs(title = "Model Comparison: Recall", x = "Model", y = "Recall") +
  theme_minimal() +
  theme(legend.position = "none")

plot4 <- ggplot(model_comparison, aes(x = Model, y = F1_Score, fill = Model))
+
  geom_bar(stat = "identity") +
  labs(title = "Model Comparison: F1 Score", x = "Model", y = "F1 Score") +
  theme_minimal() +
  theme(legend.position = "none")

plot5 <- ggplot(model_comparison, aes(x = Model, y = AUC, group = 1)) +
  geom_line(aes(color = Model), size = 1) +
  geom_point(aes(color = Model), size = 3) +
  labs(title = "Model Comparison: AUC", x = "Model", y = "AUC") +
  theme_minimal() +
  theme(legend.position = "none")

# Arrange the plots in a grid
grid.arrange(plot1, plot2, plot3, plot4, plot5, ncol = 2)
```



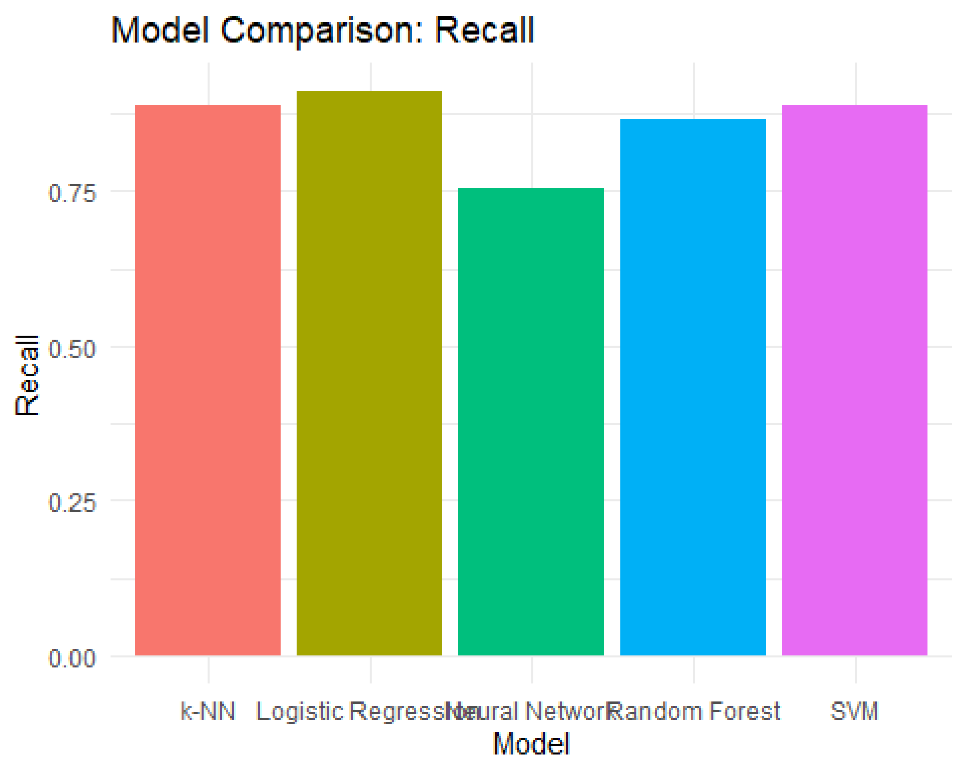
```r
print(plot1)
```



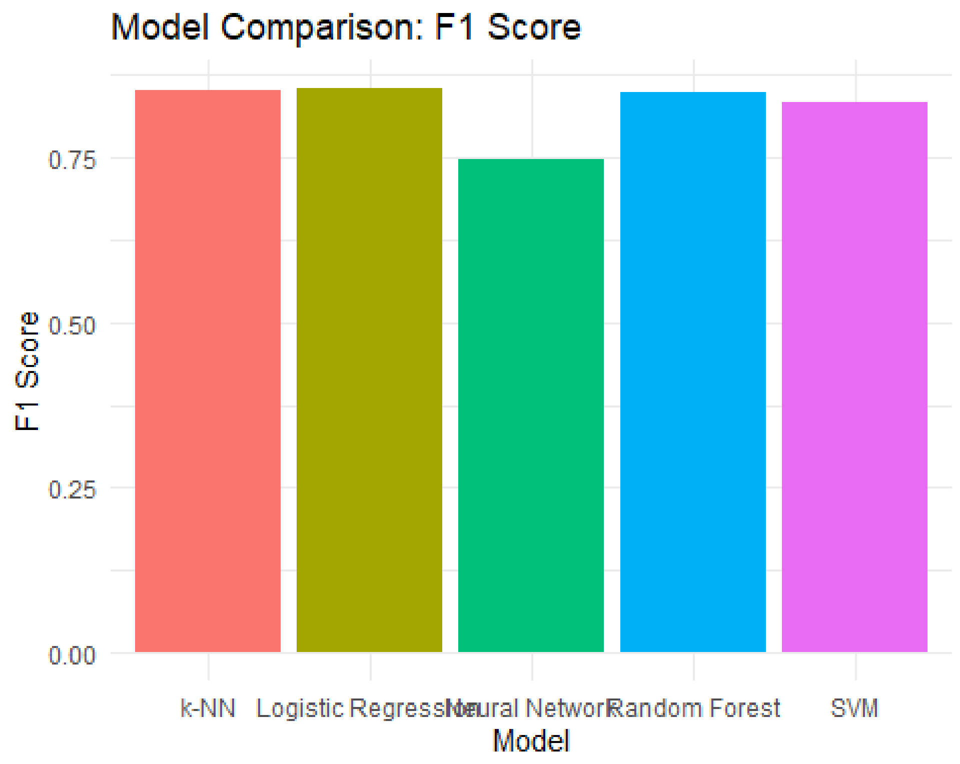
```
print(plot2)
```



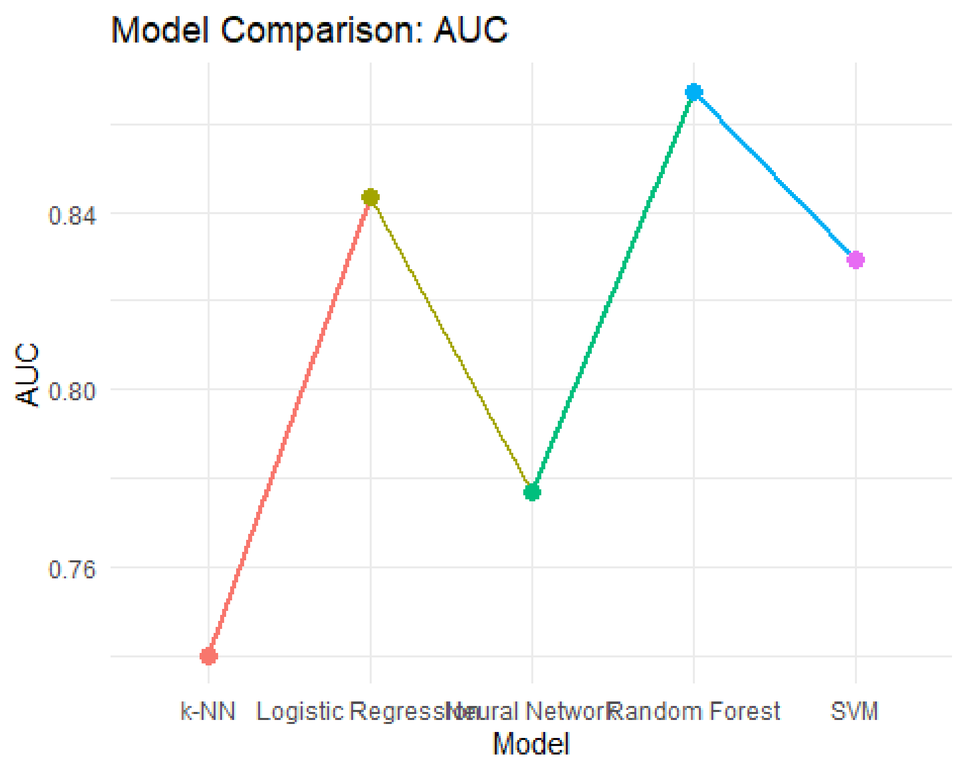
```
print(plot3)
```



```r
print(plot4)
```

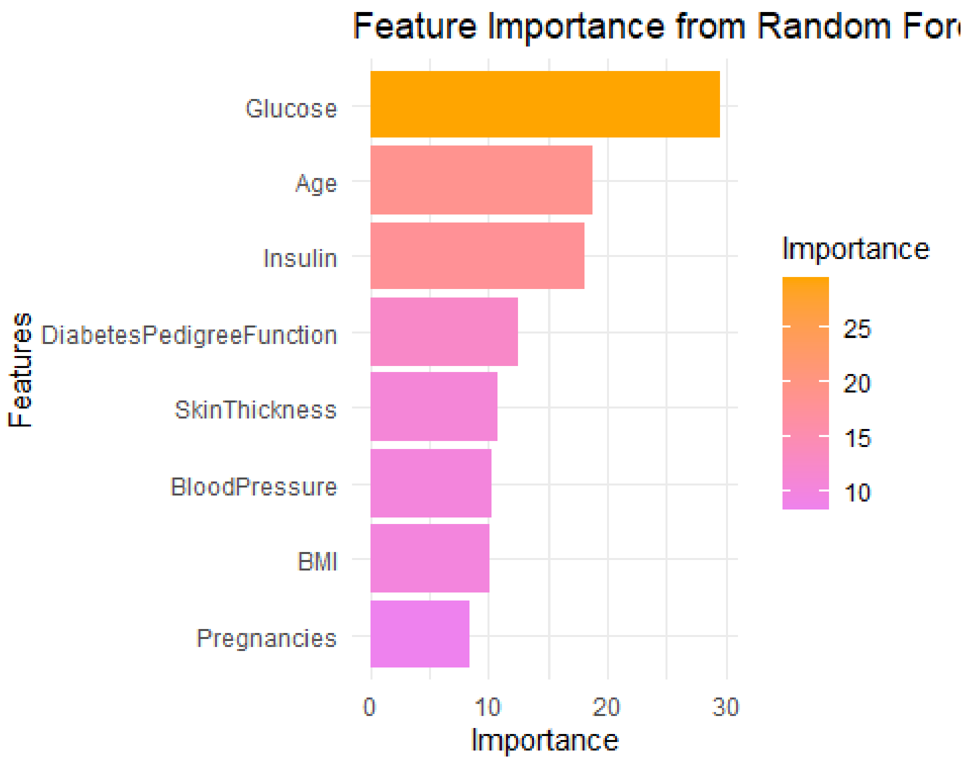


```r
print(plot5)
```



```
# Fit random forest model
rf_model <- randomForest(Outcome ~ ., data = train_data, ntree = 100, mtry =
3, importance = TRUE)

# Get feature importance
importance <- importance(rf_model)
importance_df <- data.frame(Feature = row.names(importance), Importance =
importance[, "MeanDecreaseGini"])

# Sort by importance
importance_df <- importance_df[order(importance_df$Importance, decreasing =
TRUE), ]

# Print feature importance
print(importance_df)

                                          Feature Importance
Glucose                                   Glucose  29.604150
Age                                           Age  18.760203
Insulin                                   Insulin  18.053296
DiabetesPedigreeFunction DiabetesPedigreeFunction  12.559633
SkinThickness                       SkinThickness  10.850048
BloodPressure                       BloodPressure  10.329266
BMI                                           BMI  10.173463
Pregnancies                           Pregnancies   8.348566
```

```
ggplot(importance_df, aes(x = reorder(Feature, Importance), y = Importance,
fill = Importance)) +
  geom_bar(stat = "identity") +
  coord_flip() +
  labs(title = "Feature Importance from Random Forest Model", x = "Features",
y = "Importance") +
  theme_minimal() +
  scale_fill_gradient(low = "violet", high = "orange")
```


Feature Importance from Random For
Glucose
Age
Insulin
DiabetesPedigreeFunction
SkinThickness
BloodPressure
BMI
Pregnancies
Features
Importance
Importance