



SCHOOL OF COMPUTER SCIENCE
PRACTICAL ASSESSMENT (Weightage 10%)
AUGUST 2024 SEMESTER


MODULE NAME	:	DATA MINING
MODULE CODE	:	ITS61504
DUE DATE	:	FRIDAY, 16th FEBRUARY, 2024
PLATFORM	:	MyTIMES

This paper consists of **THREE (3)** pages, inclusive of this page.

ASSIGNMENT TOPIC: “DIABETES CLASSIFICATION”

STUDENT DECLARATION

- 1. I confirm that I am aware of the University's Regulation Governing Cheating in a University Test and Assignment and of the guidance issued by the School of Computing and IT concerning plagiarism and proper academic practice, and that the assessed work now submitted is in accordance with this regulation and guidance.*
- 2. I understand that, unless already agreed with the School of Computing and IT, assessed work may not be submitted that has previously been submitted, either in whole or in part, at this or any other institution.*
- 3. I recognise that should evidence emerge that my work fails to comply with either of the above declarations, then I may be liable to proceedings under Regulation*

No.	Student Name	ID	Date	Signature	Score
1	Prabin Joshi	0358667	16 th Feb, 2024		

Marking Rubrics

Task 3: Marking Rubrics MLO3 (lecturer only)					
Question	Weight	Excellent	Good	Average	Poor
Q1	10	Understanding and evaluating successfully, different data mining techniques as per the real case scenario requirement, to complete the required analysis.	Understanding and evaluating moderately, different data mining techniques as per the real case scenario requirement, to complete the required analysis.	Understanding and evaluating majorly, different data mining techniques as per the real case scenario requirement, to complete the required analysis.	Understanding and evaluating a few, different data mining techniques as per the real case scenario requirement, to complete the required analysis.
Q2	10				
Q3	10				
Q4	10				

Contents

Marking Rubrics.....	2
Table of Figures	4
Table of Tables.....	5
Table of Abbreviations	6
1) Prepare the dataset with various pre – processing techniques.	7
2) Perform explanatory data analysis. Justify any feature selection and elimination is necessary.	9
3) Develop a predictive method or classification model to determine the factor of diabetes or no diabetes.	15
4) Evaluate the performance of the model using performance matrix and determine which factor affect the answer. Justify your answer.	16

Table of Figures

Figure 2:- Checking the Missing Values	7
Figure 3:- Balancing the Target Variable	7
Figure 4:- Feature Scaling.....	8
Figure 5:- Identify and Remove Outliers	8
Figure 6:- Understand the Dataset.....	9
Figure 7:- Summary Statistics for Numerical Variables.....	10
Figure 8:- Summary for Categorical Variable.....	10
Figure 9:- Histograms for Continuous Variables	11
Figure 10:- Barplots for Outcome Distribution.....	12
Figure 11:- Scatterplot of Glucose vs. Age with Outcome	13
Figure 12:- Correlation Matrix.....	14
Figure 13:- Build the SVM and KNN Models	15
Figure 14:- Model Comparison Results.....	16
Figure 15:- Scaled KNN's Confusion Matrix.....	17
Figure 16:- SVM's Confusion Matrix.....	17
Figure 17:- Barplots of Performance Evaluation Metrics	17

Table of Tables

Table 1:- Table of Abbreviations 6

Table of Abbreviations

KNN	k-nearest neighbors
SVM	Support Vector Machine

Table 1:- Table of Abbreviations

1) Prepare the dataset with various pre – processing techniques.

a) Checking Missing Values:

```
> # Check for missing values in the dataset
> missing_values <- colSums(is.na(diabetes_data))
> print("Missing Values Summary:")
[1] "Missing Values Summary:"
> print(missing_values)
      Pregnancies      Glucose      BloodPressure
              0              0              0
      SkinThickness      Insulin      BMI
              0              0              0
DiabetesPedigreeFunction      Age      Outcome
              0              0              0
```

Figure 1:- Checking the Missing Values

Here, the code checks for missing values, but none are found in the dataset. As a result, imputation techniques or removal strategies are not required. The absence of missing values ensures the integrity of the dataset, preventing potential issues during model training and evaluation.

In cases where there are missing values in a dataset, various techniques such as mean imputation, median imputation, or K-nearest neighbors (KNN) imputation might have been utilized. These approaches aim to preserve the dataset's integrity by substituting missing values with reasonable estimates derived from the existing data. Implementing imputation helps prevent the loss of crucial information and ensures that models can be trained on complete datasets, ultimately improving predictive accuracy.

b) Balancing the Target Variable:

```
> class_distribution <- table(diabetes_data$Outcome)
> print("Class Distribution:")
[1] "Class Distribution:"
> print(class_distribution)
      0      1
500 268
```

Figure 2:- Balancing the Target Variable

Class imbalance is a common challenge in classification tasks, where one outcome class significantly outnumbers the other. Imbalanced classes can lead to biased model performance, where the algorithm might favor the majority class. In the context of diabetes classification, the code checks the distribution of the target variable, Outcome, to identify any imbalance using `table(diabetes_data$Outcome)`. It identifies an imbalance distribution:-

- *0 representing No Diabetes: 500 instances*

- *1 representing Diabetes: 268 instances*

To address this issue, I utilized the Synthetic Minority Over-sampling Technique (SMOTE) using the “ROSE” package. This technique artificially generates synthetic samples for the minority class, creating a balanced distribution. Balancing the classes enhances the model's ability to generalize and make accurate predictions for both classes.

c) **Feature Scaling:**

```
> # Feature scaling using the preProcess function from caret
> scaling_model <- preProcess(train_data[, predictors], method = c("center", "scale"))
> # Apply the scaling transformation to both the training and test sets
> train_data_scaled <- predict(scaling_model, train_data[, predictors])
> test_data_scaled <- predict(scaling_model, test_data[, predictors])
```

Figure 3:- Feature Scaling

Feature scaling is a crucial pre-processing step, especially when working with algorithms sensitive to the scale of input features. In this dataset, feature scaling is accomplished using the preProcess function with the "center" and "scale" methods. This step is essential because features with different scales might disproportionately influence the model, leading to biased results.

By standardizing the numerical features, the algorithm can treat all features equally during the training process. The resulting scaled dataset, stored in “train_data_scaled” and “test_data_scaled”, ensures that each feature contributes proportionately to the overall predictive model. Feature scaling enhances the model's robustness and ensures that it performs optimally across all input features.

d) **Handling Outliers:**

```
> print("Number of Outliers in Each Numerical Variable:")
[1] "Number of Outliers in Each Numerical Variable:"
> print(colSums(outliers))
      Pregnancies      Glucose      BloodPressure
              4              5              45
      SkinThickness      Insulin              BMI
              1             34             19
DiabetesPedigreeFunction      Age
              29              9
> # Remove outliers from the dataset
> diabetes_data_no_outliers <- diabetes_data[!apply(outliers, 1, any), ]
> # Display information before and after removing outliers
> cat("Original dataset size:", nrow(diabetes_data), "rows\n")
Original dataset size: 768 rows
> cat("Dataset size after removing outliers:", nrow(diabetes_data_no_outliers), "rows\n")
Dataset size after removing outliers: 639 rows
```

Figure 4:- Identify and Remove Outliers

The code generates boxplots for each numerical variable in the diabetes dataset to identify outliers. The subsequent outlier detection function identifies outliers in each variable based on the interquartile range (IQR) method. The results indicate the number of outliers in each variable, and by removing these outliers, the dataset size is reduced from the original 768 rows to 639 rows.

Identifying outliers is crucial for data quality and model robustness. Outliers can significantly impact statistical analyses and machine learning models by skewing results and introducing noise. The chosen technique, using boxplots and the interquartile range (IQR) method, is effective as it provides a visual representation of the data's central tendency and spread. The IQR method identifies values significantly deviating from the majority, offering a systematic approach to flag potential outliers. By removing outliers, the dataset is refined, enhancing the reliability of subsequent analyses and model training by mitigating the influence of extreme values.

2) Perform explanatory data analysis. Justify any feature selection and elimination is necessary.

Exploratory Data Analysis (EDA) is a statistical approach that involves visualizing, summarizing, and interpreting the main characteristics of a dataset to uncover patterns, trends, and relationships, helping analysts gain insights and inform further analysis or modeling decisions. I will conduct a comprehensive analysis for this EDA part, which are given bellows:-

→ Dataset Basic Information:

```
> # Display basic information about the dataset
> str(diabetes_data)
'data.frame': 768 obs. of 9 variables:
 $ Pregnancies      : int  6 1 8 1 0 5 3 10 2 8 ...
 $ Glucose          : int  148 85 183 89 137 116 78 115 197 125 ...
 $ BloodPressure    : int  72 66 64 66 40 74 50 0 70 96 ...
 $ SkinThickness    : int  35 29 0 23 35 0 32 0 45 0 ...
 $ Insulin          : int  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              : int  50 31 32 21 33 30 26 29 53 54 ...
 $ Outcome          : int  1 0 1 0 1 0 1 0 1 1 ...
```

Figure 5:- Understand the Dataset

The dataset "diabetes_data" contains 768 observations and 9 variables. These variables include information such as the number of pregnancies, glucose levels,

blood pressure, skin thickness, insulin levels, BMI, diabetes pedigree function, age, and the outcome (1 for diabetes, 0 for non-diabetes).

Furthermore, the dataset represents diverse health-related features, both numerical and categorical, making it suitable for predictive modeling in the context of diabetes classification. The "str" function provides a concise summary, displaying the data types and structure of each variable in the dataframe.

→ Summary Statistics for Numerical Variables:

```
> print(summary_stats_numerical)
```

Pregnancies		Glucose		BloodPressure		SkinThickness		Insulin	
Min.	: 0.000	Min.	: 0.0	Min.	: 0.00	Min.	: 0.00	Min.	: 0.0
1st Qu.	: 1.000	1st Qu.	: 99.0	1st Qu.	: 62.00	1st Qu.	: 0.00	1st Qu.	: 0.0
Median	: 3.000	Median	: 117.0	Median	: 72.00	Median	: 23.00	Median	: 30.5
Mean	: 3.845	Mean	: 120.9	Mean	: 69.11	Mean	: 20.54	Mean	: 79.8
3rd Qu.	: 6.000	3rd Qu.	: 140.2	3rd Qu.	: 80.00	3rd Qu.	: 32.00	3rd Qu.	: 127.2
Max.	: 17.000	Max.	: 199.0	Max.	: 122.00	Max.	: 99.00	Max.	: 846.0

BMI		DiabetesPedigreeFunction		Age	
Min.	: 0.00	Min.	: 0.0780	Min.	: 21.00
1st Qu.	: 27.30	1st Qu.	: 0.2437	1st Qu.	: 24.00
Median	: 32.00	Median	: 0.3725	Median	: 29.00
Mean	: 31.99	Mean	: 0.4719	Mean	: 33.24
3rd Qu.	: 36.60	3rd Qu.	: 0.6262	3rd Qu.	: 41.00
Max.	: 67.10	Max.	: 2.4200	Max.	: 81.00

Figure 6:- Summary Statistics for Numerical Variables

The summary statistics for the numerical variables in the diabetes dataset reveal key insights into the distribution of health-related features. The statistics include measures such as minimum, maximum, mean, median, and quartiles for numeric variables like pregnancies, glucose levels, blood pressure, skin thickness, insulin levels, BMI, diabetes pedigree function, and age.

These descriptive statistics provide a comprehensive overview of the central tendency and variability of each numerical variable, offering valuable information for understanding the dataset's characteristics and potential patterns related to diabetes.

→ Summary for Categorical Variable:

```
> # Function to get a count and unique values summary for categorical variables including frequency percentage
> categorical_summary <- function(column_name) {
+   cat_summary <- table(diabetes_data[[column_name]])
+   unique_values <- length(unique(diabetes_data[[column_name]]))
+   total_count <- sum(cat_summary)
+   percentage <- prop.table(cat_summary) * 100 # Calculate percentage
+   cat_summary_df <- data.frame(
+     Category = names(cat_summary),
+     Count = as.numeric(cat_summary),
+     Percentage = percentage
+   )
+   print(paste("summary for", column_name, ":"))
+   print(paste("Unique Values:", unique_values))
+   print("Count summary:")
+   print(cat_summary_df)
+ }
> # Apply the updated function to the outcome variable
> categorical_summary("Outcome")
[1] "summary for Outcome : "
[1] "Unique Values: 2"
[1] "Count summary:"
  Category Count Percentage.Var1 Percentage.Freq
1         0     500           0         65.10417
2         1     268           1         34.89583
```

Figure 7:- Summary for Categorical Variable

The "categorical_summary" function is designed to provide a summary of count and unique values, including frequency percentages, for categorical variable in the diabetes dataset.

The function in this particular application is applied on the "Outcome" variable, which represents the binary classification of having diabetes (1) or not (0). According to the summary, "Outcome" has two distinct values: 500 instances of non-diabetes (0) and 268 instances of diabetes (1). The accompanying percentages illustrate the distribution, with 65.10% classifying as non-diabetes and 34.90% as diabetes.

→ **Univariate Analysis:**

- I conducts univariate analysis for continuous variables and categorical variable in the diabetes dataset through histograms and bar plots, which are shown in figure numbers 8 and 9.
- For continuous variables such as pregnancies, glucose levels, blood pressure, skin thickness, insulin, BMI, diabetes pedigree function, and age, I utilize histograms to visually represent their respective distributions. These histograms are presented in Figure Number 8.

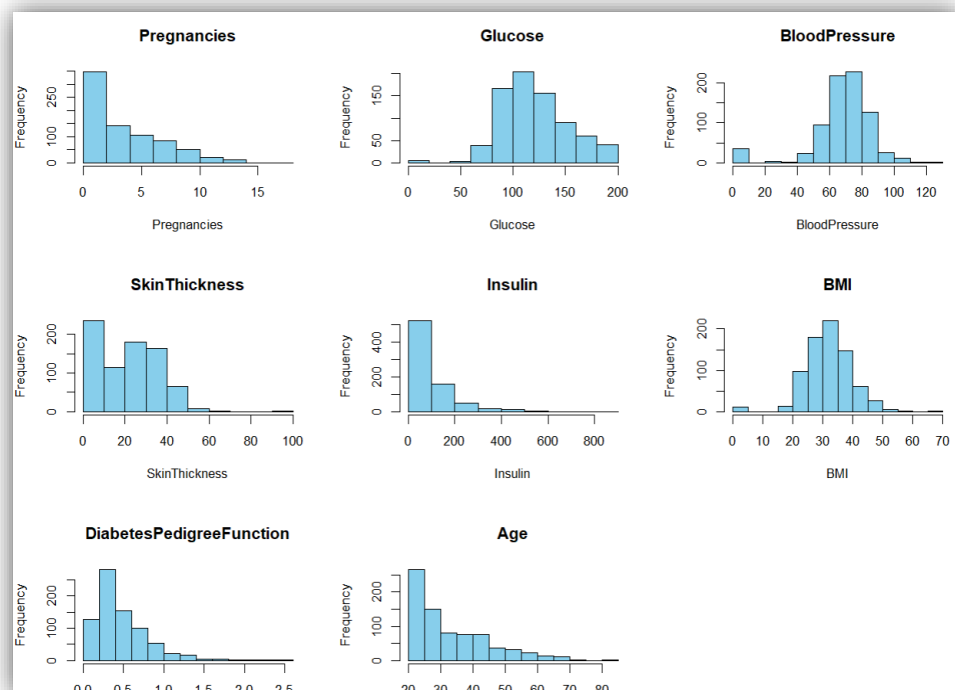


Figure 8:- Histograms for Continuous Variables

It can be clearly seen that, for "Pregnancies," a notable frequency is observed in the 0 to 5 range, declining with increasing pregnancies. Similarly, the "Glucose" histogram illustrates a higher frequency in the mid-range glucose levels. The histogram shows a peak around 100, indicating that many people have glucose levels within this range. In the "BloodPressure" histogram, the dataset showcases a significant frequency in the mid-range (around 60 to 80), suggesting a common distribution of blood pressure values. For "SkinThickness," the histogram indicates variability in the dataset, with noticeable concentrations around zero and peaks at certain thickness levels. This indicates that there is a diversity of skin thicknesses in the dataset.

The "Insulin" histogram demonstrates a right-skewed distribution, emphasizing lower insulin levels with a few instances of higher values. In the case of "BMI," the histogram shows a prevalent mid-range distribution around 25, with more individuals classified as overweight or obese. The "DiabetesPedigreeFunction" histogram illustrates a right-skewed distribution, indicating lower values are more frequent, emphasizing a lower genetic predisposition. Finally, the age histogram suggests a relatively even distribution across age groups, with a notable frequency in the middle-aged range, approximately around 40, indicating a diverse dataset with a peak in middle-aged individuals.

- To depict the distribution of categorical variable (Outcome), I generate bar plots showcasing the frequency of each category, as depicted in Figure Number 9.

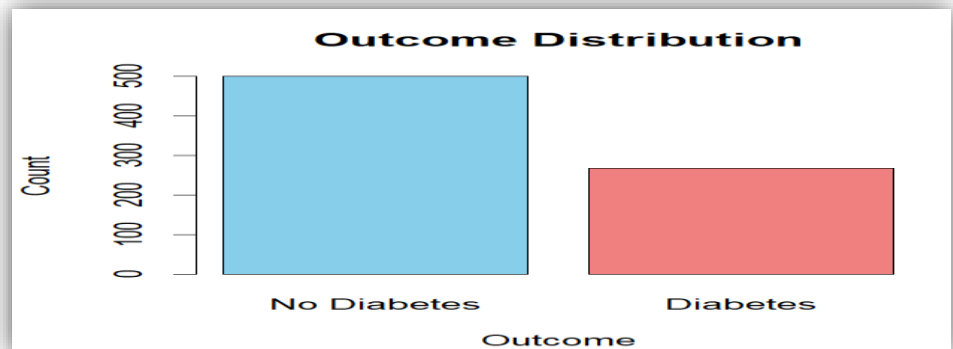


Figure 9:- Barplots for Outcome Distribution

Compared to the binary target variable "Outcome" denoting diabetes (1) or non-diabetes (0), the figure no. 9 indicates a higher count of instances labeled as "No Diabetes" in contrast to "Diabetes."

→ **Scatter Plot:**

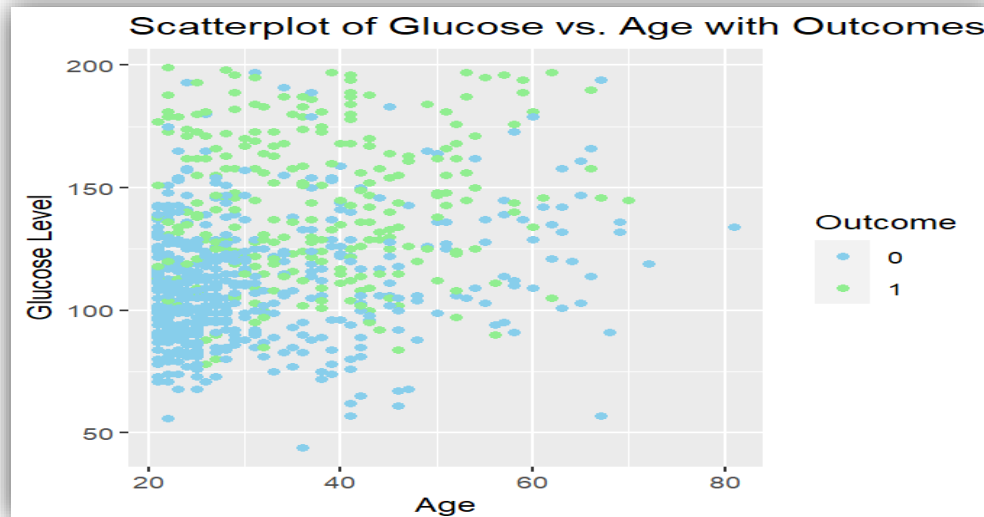


Figure 10:- Scatterplot of Glucose vs. Age with Outcome

The scatter plot visually represents individuals' data points categorized by diabetes status, using light green and sky blue to distinguish between the groups. The plot focuses on age and glucose levels, revealing a positive correlation for both groups, indicating that as age increases, glucose levels also tend to rise. Notably, the slope of the correlation is steeper for the light green category compared to the sky blue category. This suggests a potentially more rapid increase in glucose levels with advancing age for individuals with diabetes, highlighting a distinct pattern in the relationship between age and glucose levels based on diabetes status.

→ **Correlation Matrix:**

```
> print(correlation_matrix)
Pregnancies      Glucose      BloodPressure
Pregnancies      1.00000000  0.12945867    0.14128198
Glucose           0.12945867  1.00000000    0.15258959
BloodPressure     0.14128198  0.15258959    1.00000000
SkinThickness     -0.08167177  0.05732789    0.20737054
Insulin          -0.07353461  0.33135711    0.08893338
BMI              0.01768309  0.22107107    0.28180529
DiabetesPedigreeFunction -0.03352267  0.13733730    0.04126495
Age              0.54434123  0.26351432    0.23952795
Outcome          0.22189815  0.46658140    0.06506836
SkinThickness     -0.08167177 -0.07353461    0.01768309
Insulin          0.05732789  0.33135711    0.22107107
BloodPressure     0.20737054  0.08893338    0.28180529
SkinThickness     1.00000000  0.43678257    0.39257320
Insulin          0.43678257  1.00000000    0.19785906
BMI              0.39257320  0.19785906    1.00000000
DiabetesPedigreeFunction 0.18392757  0.18507093    0.14064695
Age              -0.11397026 -0.04216295    0.03624187
Outcome          0.07475223  0.13054795    0.29269466
DiabetesPedigreeFunction -0.03352267  0.54434123
Glucose           0.13733730  0.26351432
BloodPressure     0.04126495  0.23952795
SkinThickness     0.18392757 -0.11397026
Insulin          0.18507093 -0.04216295
BMI              0.14064695  0.03624187
DiabetesPedigreeFunction 1.00000000  0.03356131
Age              0.03356131  1.00000000
Outcome          0.17384407  0.23835598
```

```

Pregnancies      Outcome
Glucose           0.22189815
BloodPressure     0.46658140
SkinThickness     0.06506836
Insulin          0.07475223
BMI              0.13054795
DiabetesPedigreeFunction 0.29269466
Age              0.17384407
Outcome          0.23835598
Outcome          1.00000000
```

Figure 11:- Correlation Matrix

The correlation matrix evaluates the relationships between various features in the diabetes dataset. Key findings include a positive correlation between "Age" and both "Pregnancies" (0.54) and "Glucose" (0.26), highlighting connections between age, pregnancy occurrences, and glucose levels. Notably, "Glucose" exhibits a strong positive correlation with "Outcome" (0.47), emphasizing the significant impact of glucose levels on diabetes outcomes. Furthermore, the correlation between "BMI" and "Outcome" (0.29) indicates a notable association between body mass index and the likelihood of diabetes. "Glucose" and "BMI" exhibit a positive correlation (0.22), indicating a relationship between higher glucose levels and increased body mass index. These insights provide valuable guidance for selecting influential features in diabetes prediction and informing subsequent analytical and modeling endeavors.

3) Develop a predictive method or classification model to determine the factor of diabetes or no diabetes.

In this diabetes classification, I used two methods i.e. SVM and KNN. Both models contribute to the development of a personalized diabetes management app for improved health monitoring and prediction.

SVM and KNN:

```
# Build the KNN model on the scaled data
knn_model_scaled <- knn(
  train = train_data_scaled,
  test = test_data_scaled,
  cl = train_data[, target], # Use the original target variable
  k = 5
)

# Build the SVM model for classification
svm_model <- svm(
  formula = as.formula(paste(target, "~", paste(predictors, collapse = "+"))),
  data = train_data,
  type = "C-classification", # Change to C-classification
  kernel = "linear",
  cost = 1
)
```

Figure 12:- Build the SVM and KNN Models

Firstly, I implemented a “Support Vector Machine” model using the “svm” function. The model is constructed with a linear kernel and is specifically configured for C-classification, aiming to predict the target variable based on a set of predictors within the training dataset. The cost parameter is set to 1, influencing the balance between achieving a smooth decision boundary and accurately classifying training points.

The Support Vector Machine (SVM) model is chosen for its effectiveness in handling complex and non-linear relationships within the diabetes dataset. SVM excels in capturing intricate decision boundaries and is particularly suitable for scenarios where the data may not be linearly separable. SVM aims to find the optimal hyperplane that maximally separates data points of different classes while maximizing the margin. In the case of this classification task, where the relationship between predictors and the target variable may be complex, SVM's ability to capture intricate patterns makes it a suitable choice. Its ability to map data into a higher-dimensional space allows it to discern subtle patterns that might be missed by simpler models. The linear kernel is chosen for its simplicity and interpretability, which is crucial in a healthcare context. Moreover, SVM offers robustness against overfitting and generalizes well to unseen data. The model's capacity to handle

high-dimensional feature spaces aligns with the diverse health indicators present in the dataset, making it a powerful choice for predicting diabetes outcomes in the context of personalized health management.

Secondly, during the construction of the KNN model, the training dataset undergoes a scaling process utilizing the `preProcess` function. Subsequently, the KNN model is established based on the scaled data through the `knn` function. This algorithm classifies a given data point by assessing the class labels of its k-nearest neighbors. In this instance, the parameter k is set to 5, signifying that the model makes its classification decision by considering the majority class among the five closest neighbors.

The k-Nearest Neighbors (KNN) model is chosen for its suitability in handling personalized diabetes predictions. KNN excels in capturing local patterns within the dataset, making it effective for scenarios where similar instances share common characteristics. Given the nature of health data, KNN's ability to identify similarities between individuals with comparable health profiles aligns well with the goal of personalized recommendations. Additionally, its simplicity and ease of interpretability make it a pragmatic choice for the development of a user-friendly diabetes management app.

4) Evaluate the performance of the model using performance matrix and determine which factor affect the answer. Justify your answer.

This study utilizes the `diabetes.csv` dataset, comprising 768 observations and 9 variables. Various data preprocessing methods are employed for cleaning, and the dataset is divided into an 80:20 ratio for training and testing. The study employs KNN (K-Nearest Neighbors) and SVM (Support Vector Method) algorithms, evaluating their performance using metrics like Accuracy, F1-Score, Recall, and Precision.

```
> print(model_metrics)
      Model Accuracy Precision    Recall  F1_Score
1 KNN Scaled 0.765625 0.5333333 0.7272727 0.6153846
2      SVM 0.828125 0.6222222 0.8484848 0.7179487
```

Figure 13:- Model Comparison Results

In the figure number 13, the model comparison results indicate that the Support Vector Machine (SVM) outperforms the K-Nearest Neighbors (KNN) model in terms of accuracy, precision, recall, and F1 score. The SVM achieved an accuracy of 82.81%, demonstrating better overall predictive performance compared to the KNN model with 76.56%. Additionally, the SVM exhibits higher precision (62.22%), recall (84.85%), and F1 Score (71.79%) compared to the KNN model. These findings suggest that the SVM model is more effective in classifying diabetes outcomes based on the provided dataset.

```
> print(conf_matrix_knn_scaled)

knn_model_scaled    0    1
                   0  74  21
                   1   9  24
```

Figure 14:- Scaled KNN's Confusion Matrix

```
> print(conf_matrix_svm)

svm_predictions    0    1
                  0  78  17
                  1   5  28
```

Figure 15:- SVM's Confusion Matrix

As shown in figure numbers 14 and 15, to assess the model's efficacy in accurately distinguishing positive and negative instances, aiding in performance evaluation and error analysis, the confusion matrix is employed. When examining the confusion matrix for the Scaled K-Nearest Neighbors (KNN) model, it is observed that it accurately predicted 74 instances of class 0 and 24 instances of class 1. However, it also misclassified 21 instances of class 0 and 9 instances of class 1.

On the other hand, the Support Vector Machine (SVM) model exhibited superior performance, as indicated by the confusion matrix, with 78 correct predictions for class 0 and 28 for class 1. Despite this, the SVM model misclassified 17 instances of class 0 and 5 instances of class 1. In summary, the SVM model demonstrates higher overall classification accuracy and fewer misclassifications in comparison to the Scaled KNN model.

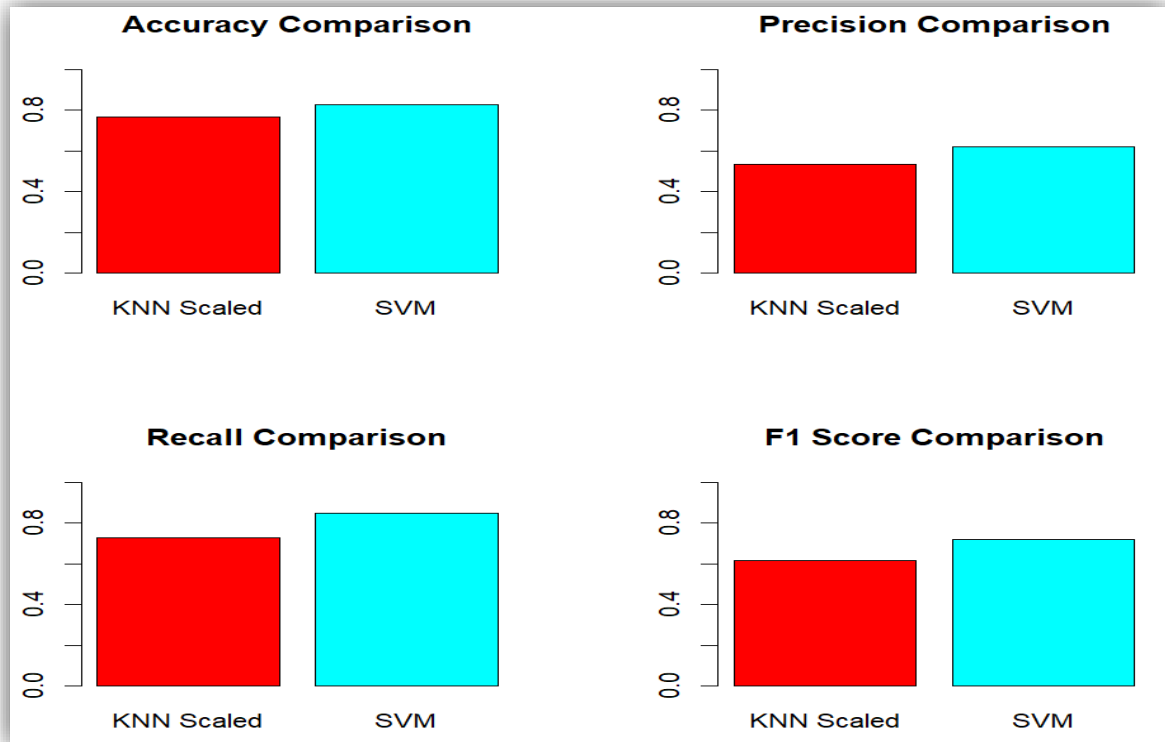


Figure 16:- Barplots of Performance Evaluation Metrics

Following the implementation of two distinct models and their subsequent evaluation, a comparative analysis of their performance is conducted using essential evaluation metrics such as Accuracy, Precision, Recall, and F1-Score. The outcomes of these metrics are then visually presented through a barplot, shown in figure number 16, to facilitate a comprehensive understanding of the models' comparative performance.

Therefore, to determine which factors affect the model's performance, an examination of the model comparison results reveals that the scaled KNN and SVM models are evaluated based on accuracy, precision, recall, and F1 score. Factors such as feature scaling, algorithm choice (KNN vs. SVM), and their specific parameter configurations contribute to the observed differences in performance metrics. The model comparison visualization helps identify which model performs better across these metrics, guiding the selection of the most suitable algorithm for the given dataset and task.