Predicting Pediatric Diabetes Using Machine Learning

Mayan Bashehab Retal Nassar

Introduction

Diabetes is a chronic metabolic disorder characterized by elevated blood sugar levels resulting from the body's inability to produce or effectively utilize insulin. It is a growing global health concern, affecting millions of people, including children worldwide [1]. Diabetes poses significant dangers to individuals, as it can lead to various complications, including heart disease, kidney failure, nerve damage, and blindness. Timely detection and management of diabetes are crucial for preventing these complications and improving the quality of life for those affected[]. Traditional methods of diabetes diagnosis rely on clinical tests and medical history, which may not capture subtle changes in an individual's physiology that indicate prediabetic conditions[1]. Using machine learning to detect diabetes has the potential to offer several benefits and advantages over traditional methods, including increased accuracy and early detection.

Our research question aims to investigate the effectiveness of machine learning algorithms in detecting diabetes in kids using a diverse range of patient health data. Diabetes is a prevalent and severe health condition with significant implications for individuals' well-being and public health. Early detection and accurate prediction of diabetes risk are essential for effective prevention and management [2].

The primary goal of our research is to employ the R language alongside data science techniques to train a dataset of diabetes health indicators obtained from Kaggle. The objective is to compare and evaluate the performance of three different machine learning models: Random Forest, Support Vector Machines (SVM), and Linear Regression. By analyzing various health indicators and applying machine learning algorithms, we aim to determine the dataset that yields the most accurate predictions for diabetes detection.

In this report, we will provide a background on the existing literature related to our research problem. Furthermore, we will provide further explanation the dataset used in our study, which comprises diabetes health indicators. To address the issue of data imbalance, we will apply techniques to balance the dataset. Subsequently, we will split the balanced dataset into training and testing sets. As mentioned, we will allocate 20% of the data for testing purposes, while the remaining 80% will be utilized for training the machine learning models.

Problem Statement and Background

Diabetes, a prevalent metabolic disorder, poses significant risks to health if undiagnosed, leading to complications such as cardiac stroke and diabetic nephropathy. The rising global incidence of diabetes has heightened the importance of early detection for maintaining a healthy life. This literature review explores the application of machine learning (ML) techniques in predictive modeling and analytics for diabetes, focusing on a study that utilizes the Pima Indian diabetes dataset and R data manipulation tools[3].

Machine learning, a computational method underpinning artificial intelligence, enables computers to learn from experience, enhancing performance for accurate predictions. In the context of healthcare, ML applications extend to disease prediction, personalized medicine, and decision support. Learning involves data gathering, preprocessing, algorithm development, testing, and optimization[3].

Ina study about predictive modeling for diabetes[4], the study utilizes a supervised ML approach on the Pima Indian diabetes dataset, emphasizing the importance of early diabetes detection. Five predictive models are developed and analyzed: Linear Kernel Support Vector Machine (SVM-linear), Radial Basis Function (RBF)

Kernel SVM, k-Nearest Neighbour (k-NN), Artificial Neural Network (ANN), and Multifactor Dimensionality Reduction (MDR). The dataset comprises medical records of female patients from the Pima Indian population, including eight risk factors. Feature selection involves using the Boruta wrapper algorithm to identify important features. The chosen ML algorithms undergo training and testing, with preprocessing including outlier removal and k-NN imputation for missing values. Predictive Model, the proposed predictive model involves preprocessing raw data, feature engineering, and supervised learning. The model's input data is derived from the Pima Indian diabetes dataset, incorporating highly correlated variables for improved outcomes.

For Results and Discussions Evaluation of the models considers various parameters, including accuracy, precision, recall, area under the curve (AUC), and F1 score. Tenfold cross-validation is employed to address overfitting and underfitting issues. The SVM-linear model achieves an accuracy of 0.89, while k-NN exhibits an accuracy of 0.88. Comprehensive parameter analysis reveals that SVM-linear and k-NN are identified as the two best models for diabetes detection, considering metrics such as precision, recall, F1 score, and AUC. Conclusion The study concludes that SVM-linear and k-NN are optimal classifiers for diabetes detection, with each excelling in specific metrics. The high AUC values for both models underscores their effectiveness. Additionally, the application of the Boruta wrapper algorithm for feature selection is highlighted, demonstrating superior results compared to manual attribute selection. The literature review emphasizes the significance of ML in diabetes prediction and the critical role of model evaluation metrics in assessing performance, especially in the context of imbalanced datasets. Future research directions could explore further refinements in predictive models and feature selection techniques[4].

I another study, [5] the impact of Artificial Intelligence (AI) on various fields, particularly in machine learning, has grown significantly over the last few decades. The study by Dagliati et al. (2017) focuses on the application of machine learning algorithms, embedded in a data mining pipeline, to predict complications associated with type 2 diabetes mellitus (T2DM) based on electronic health record (EHR) data. The authors conducted this research within the MOSAIC project, funded by the European Union, aiming to derive predictive models for T2DM complications. The research involved nearly 1,000 T2DM patients at the Istituto Clinico Scientifico Maugeri (ICSM) hospital in Italy. The predictive models focused on complications such as retinopathy, neuropathy, and nephropathy, considering different time scenarios (3, 5, and 7 years from the first hospital visit). The research utilized a four-step data mining pipeline: Center Profiling: Assessing hospital characteristics and patient care patterns. Predictive Model Targeting: Identifying modeling strategies based on center profiling and literature review. Predictive Model Construction: Developing models considering complications, time scenarios, and patient features. Predictive Model Validation: Evaluating model performance using a leave-one-out validation strategy. Predictive Models and Features: The study employed machine learning algorithms, including logistic regression (LR), naïve Bayes (NB), support vector machines (SVMs), and random forest (RF). Features considered for prediction included demographic data, clinical data from EHR, and administrative data. Missing data were handled using random forest imputation. Class Unbalance and Imputation: Class imbalance issues in the dataset, with a large number of patients without complications, were addressed through the oversampling of the minority class. Data imputation was performed using the missForest algorithm, outperforming mean or median imputation methods. For Results and Model Selection: The authors compared model performances in terms of area under the ROC curve (AUC), sensitivity, specificity, accuracy, positive predictive value (PPV), negative predictive value (NPV), and Matthews correlation coefficient (MCC). LR with rebalanced classes, based on Akaike information criterion feature selection, was selected as the preferred model for clinical practice due to its interpretability and applicability. Clinical Applications and Nomograms: LR models provided interpretable results, allowing the calculation of the probability of developing complications within specific periods. Nomograms were created to visually represent LR model results for retinopathy, neuropathy, and nephropathy, aiding clinicians in understanding individual patient risk factors. Limitations and Future Directions: The study acknowledges limitations, including the absence of certain variables like albumin-creatinine ratio and the need for further external validation. The authors emphasize the potential of predictive models in clinical decision support and the importance of incorporating individual factors for better risk stratification. To conclude, The research demonstrates the effective application of data mining and computational methods in clinical medicine. Predictive models derived from patient-specific information offer valuable insights into the risk of chronic microvascular complications in T2DM. The study advocates for integrating such models into clinical

information systems to support informed decision-making in inpatient treatment.

Data

The dataset used in this project is derived from the BRFSS survey, an annual telephone survey conducted by the CDC since 1984. It is designed to collect comprehensive data on health-related risk behaviors, chronic health issues, and the utilization of preventive treatments among Americans. The survey gathers information from over 400,000 individuals each year, making it a valuable source for analyzing health-related trends and risk factors[6].

For this report, we utilized a CSV file available on Kaggle containing data from the 2015 BRFSS survey. The data comprises responses from 441,455 participants, encompassing 330 features. These features include both direct questions posed to participants and derived variables based on their individual replies. By leveraging this rich dataset, we aimed to develop a robust predictive model for diabetes risk assessment. The dataset contains 253,680 survey responses to the CDC's BRFSS2015. The target variable Diabetes_binary has 2 classes. 0 is for no diabetes, and 1 is for prediabetes or diabetes. This dataset has 21 feature variables and is not balanced. Thus, we will perform the necessary steps to balance the data in our analysis.

The outcome variable in our project is Diabetes-binary, which is a binary value that indicates the presence or absence of diabetes. The data contains 22 variables, most of which are represented by values 0 or 1, indicating "no" or "yes" respectively. Some of the variables are continuous (numerical) while others are categorical. Specifically, continuous data in our data set are BMI, Age, and Income. Fortunately, the dataset used for this project is already clean, with all 253,680 entries present. These values altogether, when inserted in the machine learning models, will help us predict diabetes in patients.

```r
# Read the CSV file into a data frame
diabetes_clean <- read.csv("diabetes_clean.csv")

# Display the structure of the data frame
str(diabetes_clean)
```

```
## 'data.frame':    253680 obs. of  22 variables:
##  $ Diabetes_binary     : num  0 0 0 0 0 0 0 0 1 0 ...
##  $ HighBP              : num  1 0 1 1 1 1 1 1 1 0 ...
##  $ HighChol            : num  1 0 1 0 1 1 0 1 1 0 ...
##  $ CholCheck           : num  1 0 1 1 1 1 1 1 1 1 ...
##  $ BMI                 : num  40 25 28 27 24 25 30 25 30 24 ...
##  $ Smoker              : num  1 1 0 0 0 1 1 1 1 0 ...
##  $ Stroke              : num  0 0 0 0 0 0 0 0 0 0 ...
##  $ HeartDiseaseorAttack: num  0 0 0 0 0 0 0 0 1 0 ...
##  $ PhysActivity        : num  0 1 0 1 1 1 0 1 0 0 ...
##  $ Fruits              : num  0 0 1 1 1 1 0 0 1 0 ...
##  $ Veggies             : num  1 0 0 1 1 1 0 1 1 1 ...
##  $ HvyAlcoholConsump   : num  0 0 0 0 0 0 0 0 0 0 ...
##  $ AnyHealthcare       : num  1 0 1 1 1 1 1 1 1 1 ...
##  $ NoDocbcCost         : num  0 1 1 0 0 0 0 0 0 0 ...
##  $ GenHlth             : num  5 3 5 2 2 2 3 3 5 2 ...
##  $ MentHlth            : num  18 0 30 0 3 0 0 0 30 0 ...
##  $ PhysHlth            : num  15 0 30 0 0 2 14 0 30 0 ...
##  $ DiffWalk            : num  1 0 1 0 0 0 0 1 1 0 ...
##  $ Sex                 : num  0 0 0 0 0 1 0 0 0 1 ...
##  $ Age                 : num  9 7 9 11 11 10 9 11 9 8 ...
##  $ Education           : num  4 6 4 3 5 6 6 4 5 4 ...
##  $ Income              : num  3 1 8 6 4 8 7 4 1 3 ...
```

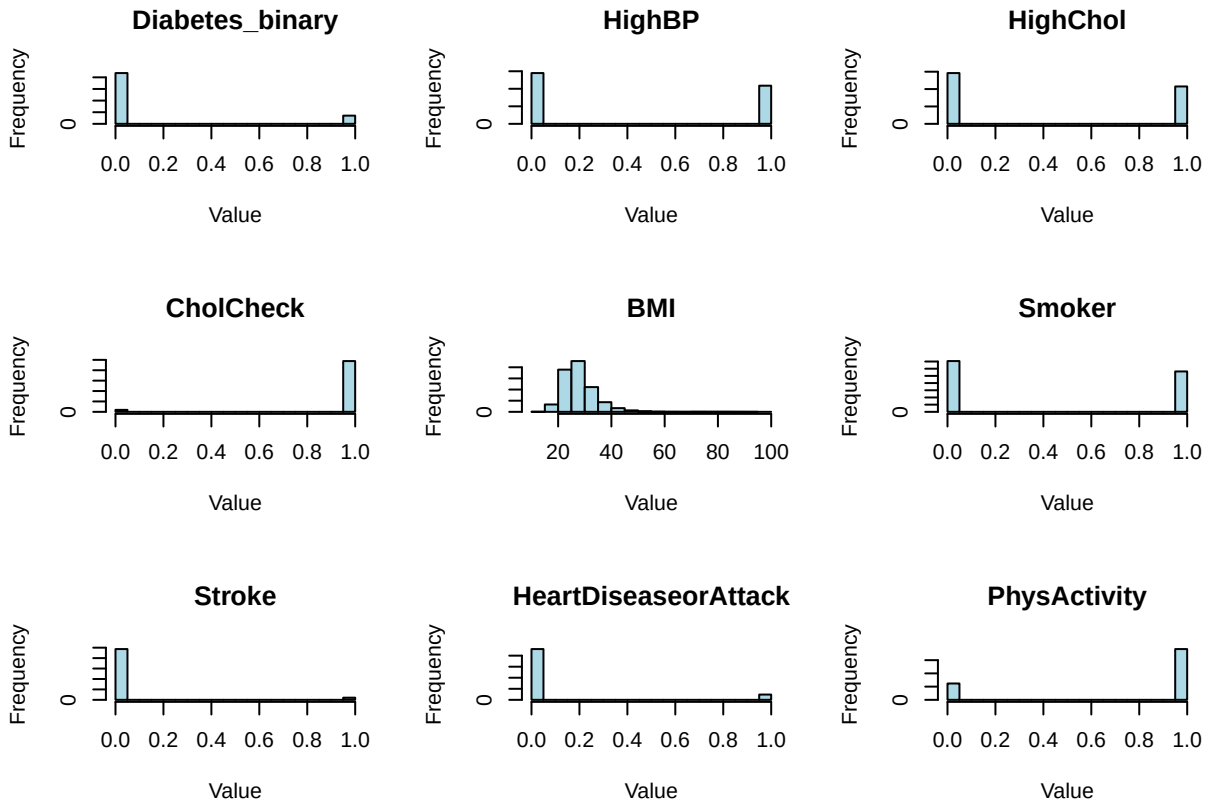
```r
# Subset the data to 10,000 entries
#subset_size <- 10000
#diabetes_clean <- diabetes_clean[sample(nrow(diabetes_clean), subset_size), ]
```

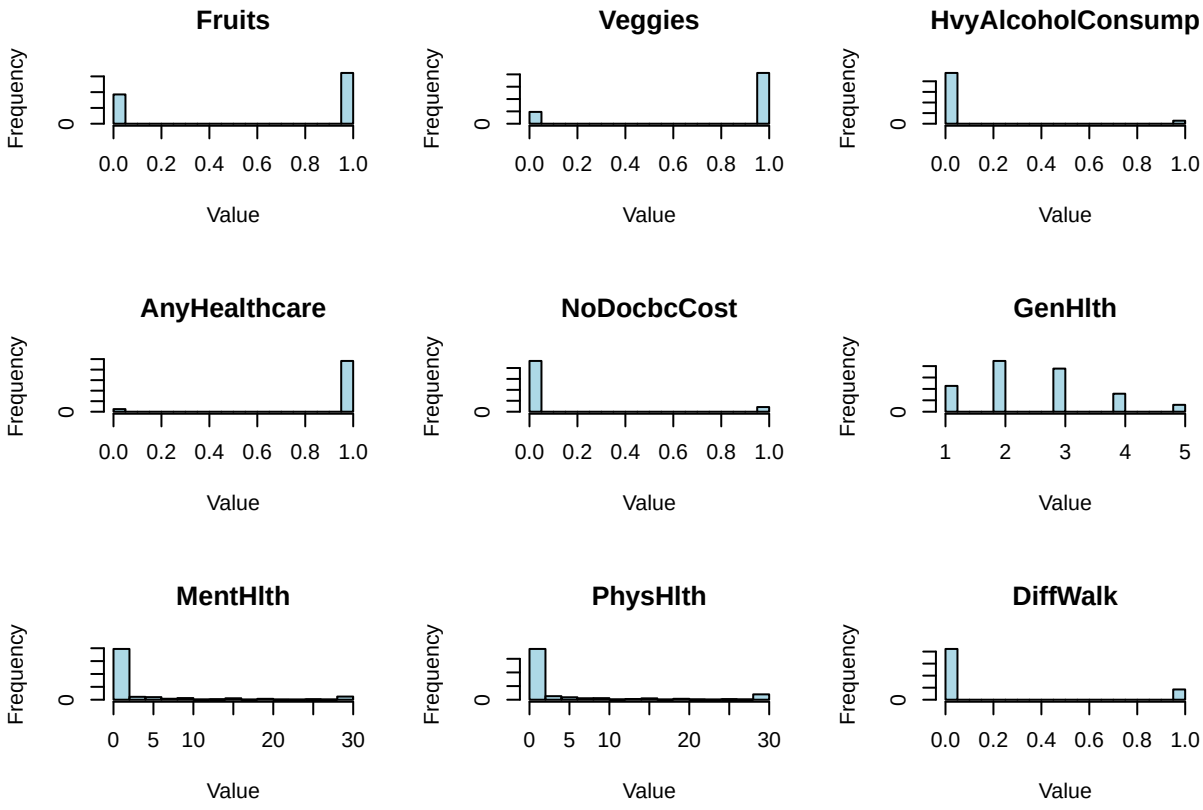
Data Visualization

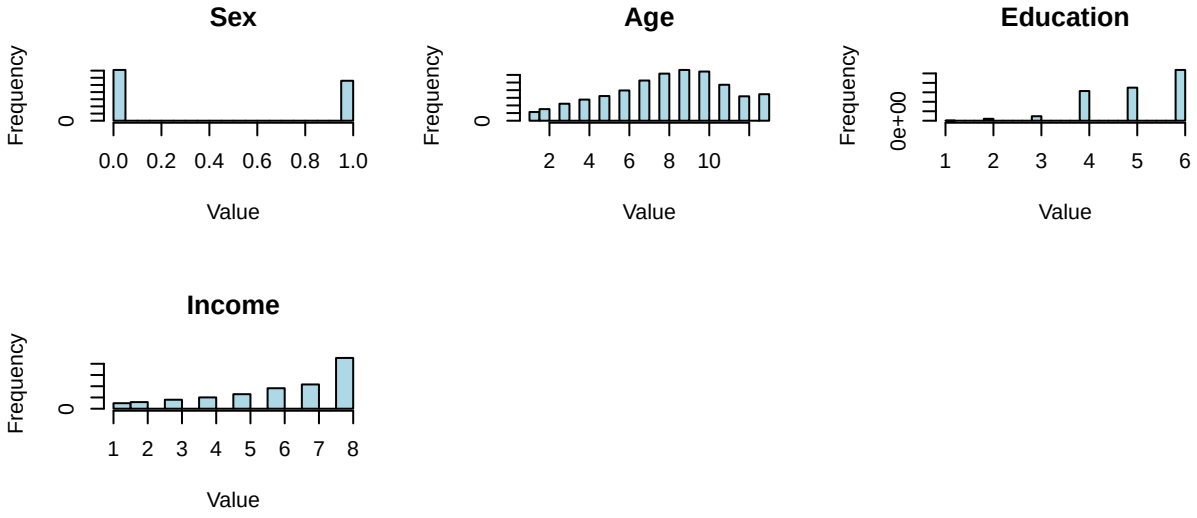
A histogram serves as a visual representation that illuminates the distribution of a dataset, offering valuable insights into its underlying characteristics[]. Prior to applying statistical analyses or machine learning algorithms, histograms provide an essential overview, enabling a comprehensive understanding of the dataset's key characteristics. This graphical depiction reveals the shape of our data distribution[7].

```r
par(mfrow=c(3, 3))  # Set the layout to 3 rows and 3 columns, adjust based on the data

# Loop through each column and create a histogram
for (col in names(diabetes_clean)) {
  hist(diabetes_clean[[col]], main=col, xlab="Value", col="lightblue", border="black")
}
```




Fruits
Veggies
HvyAlcoholConsump
AnyHealthcare
NoDocbcCost
GenHlth
MentHlth
PhysHlth
DiffWalk
Frequency
Value
0
0.0 0.2 0.4 0.6 0.8 1.0
1 2 3 4 5
0 5 10 20 30

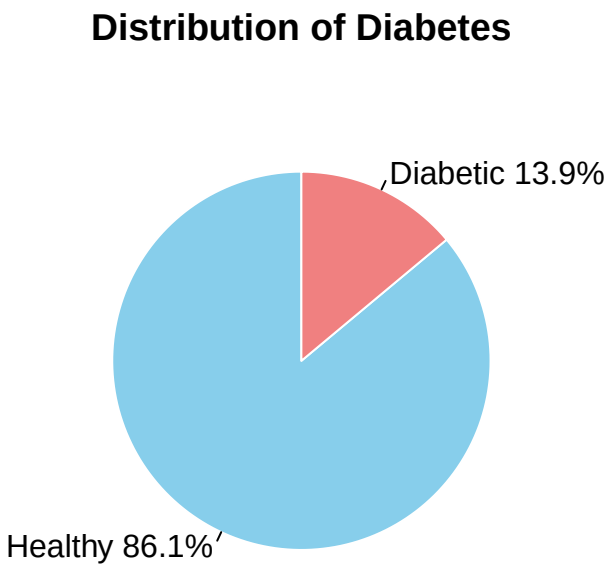


In our analysis, we observed that the dataset does not exhibit a 50-50 split between diabetic and healthy individuals, which aligns with our objective. This discrepancy is intentional and has been implemented to reflect real-world conditions. According to a report published in 2021, the prevalence of diabetes in the global population is estimated to be 10.9% [1]. By maintaining this proportion within our dataset, we aim to create a more realistic representation of the actual distribution of diabetic and healthy individuals. This approach is crucial as it enhances the validity of our machine learning model's results and ensures its effectiveness in real-life scenarios[8].

Furthermore, we have visualized the distribution of diabetic and non-diabetic individuals in our dataset using a pie diagram. The pie diagram accurately represents the exact percentages of these two groups within our dataset. This visualization provides a clear understanding of the proportion of diabetic individuals compared to non-diabetic individuals[10]. By incorporating this visual representation, we can easily grasp the relative sizes of these groups and gain insights into the dataset's composition.

```
# The outcome variable is "Diabetes_binary"
labels <- c("Healthy", "Diabetic")
counts <- table(diabetes_clean$Diabetes_binary)
percentages <- prop.table(counts) * 100  # Calculate percentages

# Creating a pie chart with percentages
pie(percentages,
    labels = paste(labels, sprintf("%.1f%%", percentages)),
    col = c("skyblue", "lightcoral"),
    main = "Distribution of Diabetes",
    init.angle = 90,
    border = "white")
```



In our analysis, a correlation matrix was performed to identify relationships and dependencies between features. This helped us to draw valuable insights into the structure, content, and relationships within the diabetes dataset, guiding into further analysis and modeling decisions[8]. We interpreted the correlation matrix map by observing the intensity of the colors. In our case, a stronger correlation was indicated by a darker shade of red, while a weaker correlation was represented by a lighter shade.

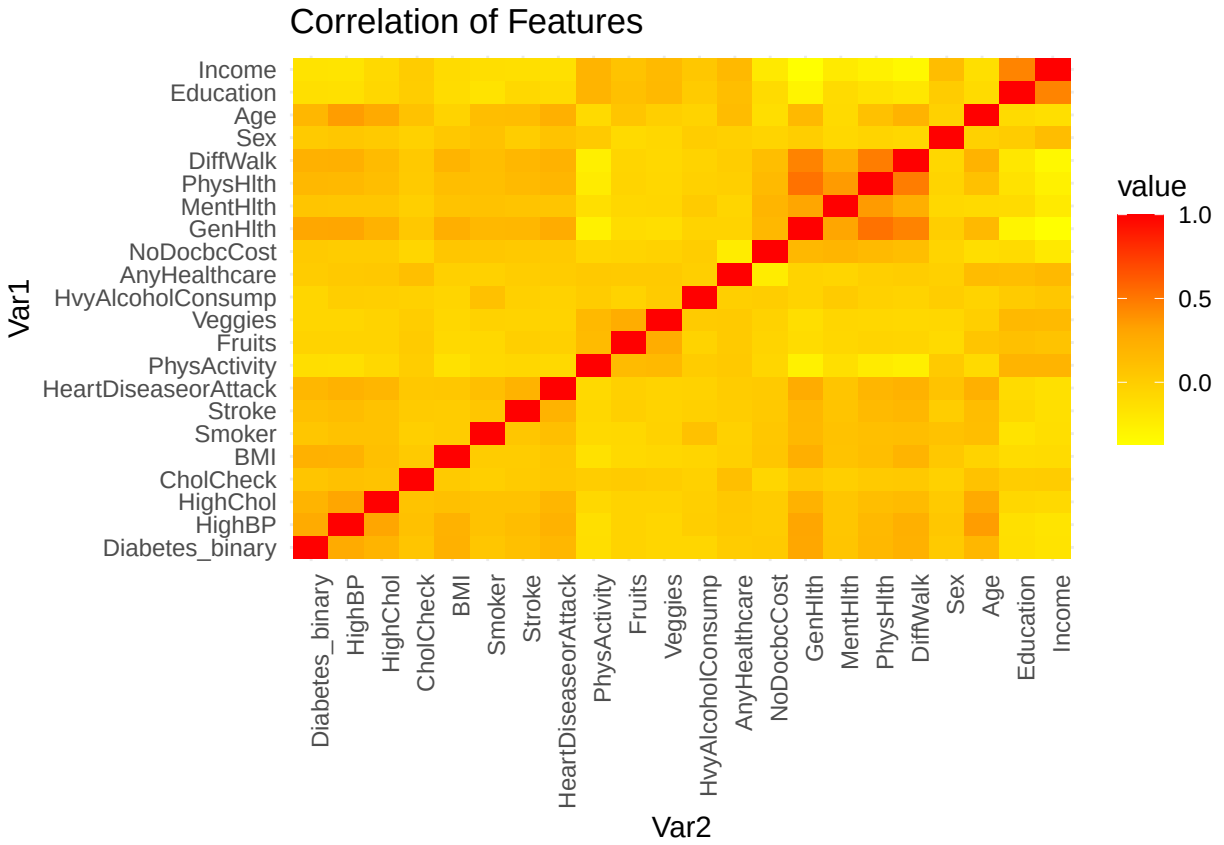
```
library(ggplot2)
library(reshape2)

# Set the figure size
options(repr.plot.width = 20, repr.plot.height = 10)

# Create a correlation matrix
cor_matrix <- cor(diabetes_clean)

# Convert the correlation matrix to a data frame
cor_df <- melt(cor_matrix)

# Plot the heatmap
ggplot(cor_df, aes(Var2, Var1, fill = value)) +
  geom_tile() +
  scale_fill_gradientn(colours = rev(colorRampPalette(c("red", "orange", "yellow"))(100))) +
  labs(title = "Correlation of Features") +
  theme_minimal() +
  theme(axis.text.x = element_text(angle = 90, hjust = 1))
```



In our correlation matrix, we can observe that variables such as general health, physical health, and difficulties walking exhibit strong correlations. Income and education seem to have a high correlation as well. This indicates that as these variables increase, they tend to be positively associated with each other.

Analysis

The provided code checks for duplicated rows in the diabetes_clean dataset and removes them. Before removing any rowb, the number of duplicates in the dataset was 24,206.

```
# Check the number of duplicated rows
sum(duplicated(diabetes_clean))
```

```
## [1] 24206
```

```
# Output the number of duplicated rows
# 24206

# Drop the duplicated rows
diabetes_clean <- diabetes_clean[!duplicated(diabetes_clean), ]

# Check the number of duplicated rows after dropping
sum(duplicated(diabetes_clean))
```

```
## [1] 0
```

This code creates a subset from the original diabetes_clean dataset, ensuring a balanced representation of non-diabetic and diabetic entries. The original dataset was large, causing computational issues and overheating.

The code filters the dataset into non-diabetic and diabetic entries. It then samples a specified percentage from each subset, creating a balanced subset with the desired number of observations for each class.

The resulting subset represents a smaller and balanced version of the original dataset, addressing computational limitations and overheating concerns while maintaining class balance for analysis.

```
#Cut the dataset

# The column representing the diabetic status is named 'Diabetes_binary'

# Load the dplyr package
library(dplyr)
```

```
##
## Attaching package: 'dplyr'

## The following objects are masked from 'package:stats':
##
##     filter, lag

## The following objects are masked from 'package:base':
##
##     intersect, setdiff, setequal, union
```

```
# Filter the original dataset to include only non-diabetic entries
non_diabetic_entries <- diabetes_clean %>% filter(Diabetes_binary == 0)

# Get the number of entries you want for the non-diabetic class (90% of 10000)
num_non_diabetic_entries <- round(0.9 * 10000)

# Sample from the non-diabetic entries to get the desired number
sampled_non_diabetic <- non_diabetic_entries[sample(nrow(non_diabetic_entries), num_non_diabetic_entrie

# Filter the original dataset to include only diabetic entries
diabetic_entries <- diabetes_clean %>% filter(Diabetes_binary == 1)

# Get the number of entries you want for the diabetic class (10% of 10000)
num_diabetic_entries <- round(0.1 * 10000)

# Sample from the diabetic entries to get the desired number
sampled_diabetic <- diabetic_entries[sample(nrow(diabetic_entries), num_diabetic_entries, replace = TRU

# Combine the sampled datasets to create the final dataset
final_dataset <- rbind(sampled_non_diabetic, sampled_diabetic)

# Shuffle the final dataset to ensure randomness
set.seed(42)  # Set seed for reproducibility
final_dataset <- final_dataset[sample(nrow(final_dataset)), ]

# Now 'final_dataset' contains 90% non-diabetic entries and 10% diabetic entries
```

Next, we use the ROSE package for oversampling to address class imbalance in the diabetes_clean dataset. It helps ensure a more balanced distribution between the classes, which is important because class imbalance can lead to biased models favoring the majority class.

The code checks the initial class distribution using table() and determines the size of the minority class. Then, it performs oversampling with a 50% oversampling proportion using the ovun.sample() function from ROSE. The resulting dataset, oversampled_data, is displayed to verify the new class distribution using table().

By performing oversampling, this code helps address class imbalance, improving the dataset's balance and reducing bias in machine learning models for more accurate predictions. The oversampling process successfully generated synthetic samples for the minority class (class 1) to balance the class distribution.

```
# Data balancing using oversampling with ROSE
library(ROSE)
```

```
## Loaded ROSE 0.0-4
```

```
# Check class distribution
table(diabetes_clean$Diabetes_binary)
```

```
##
##      0      1
## 194377  35097
```

```
# Define the target variable
target_var <- "Diabetes_binary"

# Count the number of samples in each class
class_counts <- table(diabetes_clean$Diabetes_binary)

# Find the smaller class size
minority_class_size <- min(class_counts)

# Set the proportion to oversample (adjust as needed)
oversample_proportion <- 0.5  # 50% oversampling, you can adjust this value

# Perform oversampling
oversampled_data <- ROSE::ovun.sample(as.formula(paste(target_var, "~ .")), data = diabetes_clean, meth

# Display the new class distribution
table(oversampled_data$Diabetes_binary)
```

```
##
##      0      1
## 194377 194309
```

In this code snippet, the objective is to split the dataset into two halves using a randomized approach to avoid any potential bias during model training and evaluation.

```
#Split the dataset in half
# Set the seed for reproducibility
set.seed(42)
```

```r
# Create an index vector
index <- sample(nrow(final_dataset))

# Randomize the dataset using the index
data_randomized <- final_dataset[index, ]

# Reset the row names
rownames(data_randomized) <- NULL

# Check the first few rows of the randomized dataset
head(data_randomized)
```

```
##   Diabetes_binary HighBP HighChol CholCheck BMI Smoker Stroke
## 1               0      0        0         1  51      0      0
## 2               0      0        0         1  39      0      0
## 3               0      0        1         1  27      0      0
## 4               0      1        1         1  42      0      0
## 5               0      0        0         1  25      1      0
## 6               0      1        1         1  27      0      0
##   HeartDiseaseorAttack PhysActivity Fruits Veggies HvyAlcoholConsump
## 1                    0            0      0       0                 0
## 2                    0            1      1       1                 0
## 3                    0            1      0       1                 0
## 4                    0            1      1       1                 0
## 5                    0            1      1       1                 0
## 6                    0            1      0       1                 0
##   AnyHealthcare NoDocbcCost GenHlth MentHlth PhysHlth DiffWalk Sex Age
## 1             1           0       4       30        0        0   0   1
## 2             1           0       2        0        2        0   0   1
## 3             1           0       2        0        0        0   1   4
## 4             1           0       3        0        0        0   0  10
## 5             1           0       2        3        0        0   0  10
## 6             1           0       2        5        0        0   1   8
##   Education Income
## 1         4      6
## 2         5      3
## 3         6      8
## 4         6      8
## 5         5      4
## 6         5      8
```

Next, we perform pre-processing by preparing the dataset for machine learning. We first randomizing it, then split it into training and testing sets with an 80-20 split, and finally resetting the row names for clarity.

```r
# Set the seed for reproducibility
set.seed(42)

# Calculate the number of rows for training and testing
num_rows <- nrow(data_randomized)
num_train <- round(0.8 * num_rows)
num_test <- num_rows - num_train
```

```r
# Split the data into training and testing sets
train_data <- data_randomized[1:num_train, ]
test_data <- data_randomized[(num_train + 1):num_rows, ]

# Reset the row names
rownames(train_data) <- NULL
rownames(test_data) <- NULL

# Check the number of rows in the training and testing sets
nrow(train_data)
```

```
## [1] 8000
```

```r
nrow(test_data)
```

```
## [1] 2000
```

Training the Data

In this R code, the caret library is loaded, and the seed is set to 100 for reproducibility. The target variable, binary outcomes for diabetes, is converted into a factor. The control parameters for 10-fold cross-validation are defined using trainControl, specifying the method as "cv" for 10-fold cross-validation. The evaluation metric is set to "Accuracy" to measure correct classifications during cross-validation. These steps set up the groundwork for robust machine learning model training and evaluation in the context of binary classification for diabetes prediction.

```r
#Set the control
library(caret)
```

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

```r
# Set the seed for reproducibility
set.seed(100)

# Convert the target variable to a factor with two levels
train_data$Diabetes_binary <- factor(train_data$Diabetes_binary)

# Define the control parameters for 10-fold cross-validation
control <- trainControl(method = "cv", number = 10)

# Define the metric for evaluation
metric <- "Accuracy"
```

Next, we start training the model using three different machine learning algoorithms: Random Forest, Support Vector Machines (SVM), and Linear Regression.

```r
# Train Model using LDA
model_LDA <- train(Diabetes_binary ~ ., data = train_data, method = 'lda', trControl = control, metric =
model_LDA
```

```
## Linear Discriminant Analysis
##
## 8000 samples
##   21 predictor
##    2 classes: '0', '1'
##
## No pre-processing
## Resampling: Cross-Validated (10 fold)
## Summary of sample sizes: 7200, 7199, 7200, 7200, 7201, 7201, ...
## Resampling results:
##
##   Accuracy   Kappa
##   0.8937504  0.1505438
```

```
# Train Model using CART
model_CART <- train(Diabetes_binary ~ ., data = train_data, method = 'rpart', trControl = control, metri
model_CART
```

```
## CART
##
## 8000 samples
##   21 predictor
##    2 classes: '0', '1'
##
## No pre-processing
## Resampling: Cross-Validated (10 fold)
## Summary of sample sizes: 7200, 7200, 7200, 7200, 7200, 7200, ...
## Resampling results across tuning parameters:
##
##   cp           Accuracy   Kappa
##   0.002835156  0.8960011  0.07454390
##   0.003037667  0.8960011  0.07454390
##   0.003341434  0.8962511  0.07377408
##
## Accuracy was used to select the optimal model using the largest value.
## The final value used for the model was cp = 0.003341434.
```

```
# Train Model using kNN
model_KNN <- train(Diabetes_binary ~ ., data = train_data, method = 'knn', trControl = control, metric =
model_KNN
```

```
## k-Nearest Neighbors
##
## 8000 samples
##   21 predictor
##    2 classes: '0', '1'
##
## No pre-processing
## Resampling: Cross-Validated (10 fold)
## Summary of sample sizes: 7199, 7200, 7200, 7200, 7200, 7199, ...
## Resampling results across tuning parameters:
##
##   k  Accuracy   Kappa
```

```
##   5  0.8851257  0.07069817
##   7  0.8895019  0.05716245
##   9  0.8927510  0.05445794
##
## Accuracy was used to select the optimal model using the largest value.
## The final value used for the model was k = 9.
```

```
# Train Model using SVM
model_SVM <- train(Diabetes_binary ~ ., data = train_data, method = 'svmRadial', trControl = control, me
model_SVM
```

```
## Support Vector Machines with Radial Basis Function Kernel
##
## 8000 samples
##   21 predictor
##    2 classes: '0', '1'
##
## No pre-processing
## Resampling: Cross-Validated (10 fold)
## Summary of sample sizes: 7201, 7200, 7201, 7200, 7200, 7199, ...
## Resampling results across tuning parameters:
##
##   C     Accuracy   Kappa
##   0.25  0.8971252  0.00000000
##   0.50  0.8971252  0.00000000
##   1.00  0.8977504  0.03305404
##
## Tuning parameter 'sigma' was held constant at a value of 0.0368181
## Accuracy was used to select the optimal model using the largest value.
## The final values used for the model were sigma = 0.0368181 and C = 1.
```

```
# Train Model using Random Forest
model_RF <- train(Diabetes_binary ~ ., data = train_data, method = 'rf', trControl = control, metric = m
model_RF
```

```
## Random Forest
##
## 8000 samples
##   21 predictor
##    2 classes: '0', '1'
##
## No pre-processing
## Resampling: Cross-Validated (10 fold)
## Summary of sample sizes: 7200, 7200, 7200, 7200, 7200, 7199, ...
## Resampling results across tuning parameters:
##
##   mtry  Accuracy   Kappa
##    2    0.8986250  0.02747132
##   11    0.8966247  0.15920335
##   21    0.8937500  0.15940404
##
## Accuracy was used to select the optimal model using the largest value.
## The final value used for the model was mtry = 2.
```

Overall, the models achieve accuracies ranging from approximately 89.4% to 90.0% with each run. The Random Forest model performs slightly better than the others, with the highest accuracy and a relatively higher kappa value indicating better agreement beyond chance. In summary, the models demonstrate reasonable accuracy in predicting diabetes, but they have varying performance in terms of sensitivity, specificity, positive predictive value, and negative predictive value.

```
model_compare <- resamples(list(LDA=model_LDA, CART= model_CART, KNN=model_CART, SVM=model_SVM, RF=model
summary(model_compare)
```

```
##
## Call:
## summary.resamples(object = model_compare)
##
## Models: LDA, CART, KNN, SVM, RF
## Number of resamples: 10
##
## Accuracy
##           Min.   1st Qu.    Median      Mean   3rd Qu.      Max. NA's
## LDA  0.8825000 0.8869815 0.8948686 0.8937504 0.8997811 0.9037500    0
## CART 0.8888889 0.8921875 0.8974359 0.8962511 0.8984375 0.9062500    0
## KNN  0.8888889 0.8921875 0.8974359 0.8962511 0.8984375 0.9062500    0
## SVM  0.8937500 0.8975000 0.8975640 0.8977504 0.8987183 0.9011264    0
## RF   0.8962500 0.8975000 0.8986866 0.8986250 0.8997191 0.9012500    0
##
## Kappa
##            Min.     1st Qu.     Median       Mean    3rd Qu.       Max. NA's
## LDA  0.08982984 0.106291565 0.16882755 0.15054379 0.17927779 0.21707414    0
## CART 0.00000000 0.014207819 0.07950232 0.07377408 0.11217073 0.17350818    0
## KNN  0.00000000 0.014207819 0.07950232 0.07377408 0.11217073 0.17350818    0
## SVM  0.01357781 0.019021414 0.02950486 0.03305404 0.03997741 0.06380612    0
## RF   0.00000000 0.005348032 0.02167854 0.02747132 0.04280811 0.06381466    0
```

Results

Overall, the models achieve accuracies ranging from approximately 89.4% to 90.0%. The Random Forest model performs slightly better than the others, with the highest accuracy and a relatively higher kappa value indicating better agreement beyond chance. In summary, the models demonstrate reasonable accuracy in predicting diabetes, but they have varying performance in terms of sensitivity, specificity, positive predictive value, and negative predictive value.

```
# Convert Diabetes_binary to factor with the same levels as in training data
test_data$Diabetes_binary <- factor(test_data$Diabetes_binary, levels = levels(train_data$Diabetes_binar

# Predictions for LDA
predicted_lda <- predict(model_LDA, test_data)
confusionMatrix(reference = test_data$Diabetes_binary, predicted_lda)
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction    0    1
##          0 1792  152
```

```
##          1   31   25
##
##                Accuracy : 0.9085
##                  95% CI : (0.895, 0.9208)
##     No Information Rate : 0.9115
##     P-Value [Acc > NIR] : 0.6984
##
##                   Kappa : 0.1797
##
##  Mcnemar's Test P-Value : <2e-16
##
##             Sensitivity : 0.9830
##             Specificity : 0.1412
##          Pos Pred Value : 0.9218
##          Neg Pred Value : 0.4464
##              Prevalence : 0.9115
##          Detection Rate : 0.8960
##    Detection Prevalence : 0.9720
##       Balanced Accuracy : 0.5621
##
##        'Positive' Class : 0
##
```

```r
# Predictions for KNN
predicted_knn <- predict(model_KNN, test_data)
confusionMatrix(reference = test_data$Diabetes_binary, predicted_knn)
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction    0    1
##          0 1795  168
##          1   28    9
##
##                Accuracy : 0.902
##                  95% CI : (0.8881, 0.9147)
##     No Information Rate : 0.9115
##     P-Value [Acc > NIR] : 0.9359
##
##                   Kappa : 0.0552
##
##  Mcnemar's Test P-Value : <2e-16
##
##             Sensitivity : 0.98464
##             Specificity : 0.05085
##          Pos Pred Value : 0.91442
##          Neg Pred Value : 0.24324
##              Prevalence : 0.91150
##          Detection Rate : 0.89750
##    Detection Prevalence : 0.98150
##       Balanced Accuracy : 0.51774
##
##        'Positive' Class : 0
##
```

```r
# Predictions for CART
predicted_cart <- predict(model_CART, test_data)
confusionMatrix(reference = test_data$Diabetes_binary, predicted_cart)
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction    0    1
##          0 1823  177
##          1    0    0
##
##                Accuracy : 0.9115
##                  95% CI : (0.8982, 0.9236)
##     No Information Rate : 0.9115
##     P-Value [Acc > NIR] : 0.52
##
##                   Kappa : 0
##
##  Mcnemar's Test P-Value : <2e-16
##
##             Sensitivity : 1.0000
##             Specificity : 0.0000
##          Pos Pred Value : 0.9115
##          Neg Pred Value :    NaN
##              Prevalence : 0.9115
##          Detection Rate : 0.9115
##    Detection Prevalence : 1.0000
##       Balanced Accuracy : 0.5000
##
##        'Positive' Class : 0
##
```

```r
# Predictions for SVM
predicted_svm <- predict(model_SVM, test_data)
confusionMatrix(reference = test_data$Diabetes_binary, predicted_svm)
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction    0    1
##          0 1821  173
##          1    2    4
##
##                Accuracy : 0.9125
##                  95% CI : (0.8993, 0.9245)
##     No Information Rate : 0.9115
##     P-Value [Acc > NIR] : 0.4572
##
##                   Kappa : 0.0381
##
##  Mcnemar's Test P-Value : <2e-16
##
##             Sensitivity : 0.9989
```

```
##             Specificity : 0.0226
##          Pos Pred Value : 0.9132
##          Neg Pred Value : 0.6667
##              Prevalence : 0.9115
##          Detection Rate : 0.9105
##    Detection Prevalence : 0.9970
##       Balanced Accuracy : 0.5108
##
##        'Positive' Class : 0
##
```

```r
# Predictions for RF
predicted_rf <- predict(model_RF, test_data)
confusionMatrix(reference = test_data$Diabetes_binary, predicted_rf)
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction    0    1
##          0 1823  175
##          1    0    2
##
##                Accuracy : 0.9125
##                  95% CI : (0.8993, 0.9245)
##     No Information Rate : 0.9115
##     P-Value [Acc > NIR] : 0.4572
##
##                   Kappa : 0.0204
##
##  Mcnemar's Test P-Value : <2e-16
##
##             Sensitivity : 1.0000
##             Specificity : 0.0113
##          Pos Pred Value : 0.9124
##          Neg Pred Value : 1.0000
##              Prevalence : 0.9115
##          Detection Rate : 0.9115
##    Detection Prevalence : 0.9990
##       Balanced Accuracy : 0.5056
##
##        'Positive' Class : 0
##
```

Discussion and conclusion

All models have similar accuracies ranging from approximately 90.0% to 91.0%. Furthermore, Sensitivity (True Positive Rate) measures the ability to correctly identify positive cases (diabetes). LDA, KNN, CART and RF have relatively high sensitivity values, indicating a good ability to detect positive cases. Pos Pred Value (Positive Predictive Value) represents the proportion of true positive predictions among positive predictions made by the model. LDA, KNN, and RF have higher values, indicating a higher proportion of correct positive predictions. Neg Pred Value (Negative Predictive Value) represents the proportion of true negative predictions among negative predictions made by the model. SVM has a relatively low value, indicating a lower proportion of correct negative predictions.

In summary, all models exhibit similar overall accuracy, but there are trade-offs between sensitivity and specificity. The SVM model has the highest sensitivity but the lowest specificity. The choice of the best machine learning model depends on the specific goals and requirements of the application. If sensitivity is crucial, the SVM model may be preferred, but if a balance between sensitivity and specificity is desired, other models like CART or Random Forest could be more suitable.

References

[1]Lai, H., Huang, H., Keshavjee, K., Guergachi, A., & Gao, X. (2019, October 15). Predictive models for diabetes mellitus using machine learning techniques. BMC Endocrine Disorders. https://doi.org/10.1186/s12902-019-0436-6

[2] Kaur, H., & Kumari, V. (2020, July 28). Predictive modelling and analytics for diabetes using a machine learning approach. Applied Computing and Informatics. https://doi.org/10.1016/j.aci.2018.12.004

[3] Wee, B. F., Sivakumar, S., Lim, K. H., Wong, W., & Juwono, F. H. (2023, August 10). Diabetes detection based on machine learning and deep learning approaches. Multimedia Tools and Applications. https://doi.org/10.1007/s11042-023-16407-5

[4] Dinh, A. Q., Miertschin, S., Young, A., & Mohanty, S. D. (2019, November 6). A data-driven approach to predicting diabetes and cardiovascular disease with machine learning. BMC Medical Informatics and Decision Making. https://doi.org/10.1186/s12911-019-0918-5

[5] Ahmad, H. F., Mukhtar, H., Alaqail, H., Seliaman, M. E., & Alhumam, A. (2021, January 27). Investigating Health-Related Features and Their Impact on the Prediction of Diabetes Using Machine Learning. Applied Sciences. https://doi.org/10.3390/app11031173

[6] Prediction of Diabetes Using Machine Learning Algorithms in Healthcare. (2018, September 1). IEEE Conference Publication | IEEE Xplore. https://ieeexplore.ieee.org/abstract/document/8748992

[7] Mujumdar, A., & Vaidehi, V. (2019, January 1). Diabetes Prediction using Machine Learning Algorithms. Procedia Computer Science. https://doi.org/10.1016/j.procs.2020.01.047

[8] Farajollahi, B., Mehmannavaz, M., Mehrjoo, H., Moghbeli, F., & Sayadi, M. J. (2021, March 4). Diabetes Diagnosis Using Machine Learning. Frontiers in Health Informatics. https://doi.org/10.30699/fhi.v10i1.267

[9] Birjais, R., Mourya, A. K., Chauhan, R., & Kaur, H. (2019, August 28). Prediction and diagnosis of future diabetes risk: a machine learning approach. SN Applied Sciences. https://doi.org/10.1007/s42452-019-1117-9

[10] Ameena, R. R., & Ashadevi, B. (2020, January 1). Predictive analysis of diabetic women patients using R. Elsevier eBooks. https://doi.org/10.1016/b978-0-12-819779-0.00006-x

[11] K. (2020, April 30). Diabetes Prediction using Machine Learning Techniques. https://helixscientific.pub/index.php/Home/article/view/123