Classifying the Onset of Diabetes Using Medical and Demographic Patient Data

2025-06-18

Table of Contents

# Overview

## Setup Environment

## Dataset Overview and Objective

The **Diabetes Prediction** Dataset is a comprehensive collection of medical and demographic records aimed at identifying the likelihood of diabetes in patients. Each observation in the dataset includes both physiological and lifestyle-related features, along with a binary outcome indicating whether the individual is diabetic (positive) or not (negative).

Key variables in the dataset include:

Age: The age of the patient Gender: Male or Female Body Mass Index (BMI): A measure of body fat based on height and weight Hypertension: Indicates whether the patient has high blood pressure (1 = Yes, 0 = No) Heart Disease: Indicates the presence of cardiovascular disease (1 = Yes, 0 = No) Smoking History: Categorical variable showing smoking habits (e.g., never, current, former) HbA1c Level: Average blood sugar level over the past 2–3 months, measured in percentage Blood Glucose Level: Instantaneous measure of blood sugar at the time of recording Diabetes Status: Target variable indicating whether the patient is diabetic (positive) or not (negative) The primary objective of this analysis is to develop a predictive classification model that determines whether an individual is likely to have diabetes based on their medical and lifestyle attributes.

# Data Exploration

## Load Data

# Clear everything  
rm(list = ls())  
gc()

## used (Mb) gc trigger (Mb) max used (Mb)  
## Ncells 2752886 147.1 4887328 261.1 4887328 261.1  
## Vcells 4465936 34.1 10146329 77.5 8145850 62.2

# Load only essential packages  
library(ggplot2)  
library(dplyr)  
  
# Load data  
df <- read.csv("diabetes\_prediction\_dataset.csv", stringsAsFactors = FALSE)  
  
# View first few rows  
head(df)

## gender age hypertension heart\_disease smoking\_history bmi HbA1c\_level  
## 1 Female 80 0 1 never 25.19 6.6  
## 2 Female 54 0 0 No Info 27.32 6.6  
## 3 Male 28 0 0 never 27.32 5.7  
## 4 Female 36 0 0 current 23.45 5.0  
## 5 Male 76 1 1 current 20.14 4.8  
## 6 Female 20 0 0 never 27.32 6.6  
## blood\_glucose\_level diabetes  
## 1 140 0  
## 2 80 0  
## 3 158 0  
## 4 155 0  
## 5 155 0  
## 6 85 0

# View structure and summary  
summary(df)

## gender age hypertension heart\_disease   
## Length:100000 Min. : 0.08 Min. :0.00000 Min. :0.00000   
## Class :character 1st Qu.:24.00 1st Qu.:0.00000 1st Qu.:0.00000   
## Mode :character Median :43.00 Median :0.00000 Median :0.00000   
## Mean :41.89 Mean :0.07485 Mean :0.03942   
## 3rd Qu.:60.00 3rd Qu.:0.00000 3rd Qu.:0.00000   
## Max. :80.00 Max. :1.00000 Max. :1.00000   
## smoking\_history bmi HbA1c\_level blood\_glucose\_level  
## Length:100000 Min. :10.01 Min. :3.500 Min. : 80.0   
## Class :character 1st Qu.:23.63 1st Qu.:4.800 1st Qu.:100.0   
## Mode :character Median :27.32 Median :5.800 Median :140.0   
## Mean :27.32 Mean :5.528 Mean :138.1   
## 3rd Qu.:29.58 3rd Qu.:6.200 3rd Qu.:159.0   
## Max. :95.69 Max. :9.000 Max. :300.0   
## diabetes   
## Min. :0.000   
## 1st Qu.:0.000   
## Median :0.000   
## Mean :0.085   
## 3rd Qu.:0.000   
## Max. :1.000

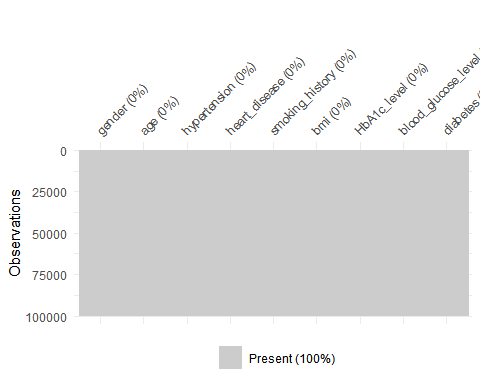
## Data Preprocessing

### Missing Values

# Check for missing values  
sum(is.na(df))

## [1] 0

vis\_miss(df) # Visualize missing data



### Encode Categorical Variables

Categorical variables were converted to factors for better readability for EDA purpose. Binary values (0 and 1) for hypertension, heart disease, and diabetes were relabeled as “No”/“Yes” or “Negative”/“Positive” to improve interpretability.

# Convert to proper types -> 0 and 1 to "yes" and "no"  
df <- df %>%  
 mutate(  
 gender = as.factor(gender),  
 smoking\_history = as.factor(smoking\_history),  
 hypertension = factor(hypertension, levels = c(0,1), labels = c("No", "Yes")),  
 heart\_disease = factor(heart\_disease, levels = c(0,1), labels = c("No", "Yes")),  
 diabetes = factor(diabetes, levels = c(0,1), labels = c("Negative", "Positive"))  
 )  
  
# Check "No Info" in smoking history  
df %>%   
 count(smoking\_history) %>%   
 mutate(prop = n/sum(n))

## smoking\_history n prop  
## 1 current 9286 0.09286  
## 2 ever 4004 0.04004  
## 3 former 9352 0.09352  
## 4 never 35095 0.35095  
## 5 No Info 35816 0.35816  
## 6 not current 6447 0.06447

### Feature Engineering

Smoking history was cleaned by renaming “No Info” to “Unknown” and merging similar categories. “Ever,” “former,” and “not current” became “Former” (19.8%), while “Never” (35.1%) and “Current” (9.3%) were kept unchanged.

# Consolidate smoking history categories  
df <- df %>%  
 mutate(smoking\_history = factor(smoking\_history)) %>%  
 mutate(  
 smoking\_history = fct\_collapse(smoking\_history,  
 "Never" = "never",  
 "Former" = c("ever", "former", "not current"),  
 "Current" = "current",  
 "Unknown" = "No Info"  
 )  
 )  
  
# Check "No Info" in smoking history  
df%>%   
 count(smoking\_history) %>%   
 mutate(prop = n/sum(n))

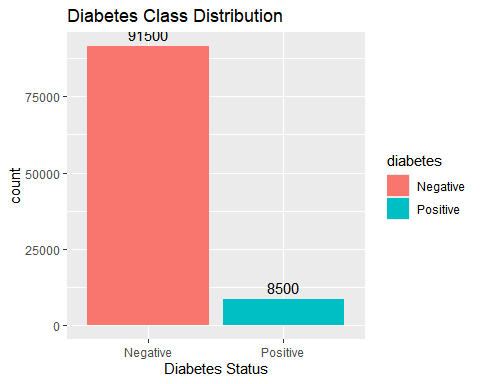
## smoking\_history n prop  
## 1 Current 9286 0.09286  
## 2 Former 19803 0.19803  
## 3 Never 35095 0.35095  
## 4 Unknown 35816 0.35816

# Exploratory Data Analysis

## Diabetes (Response) Class Distribution

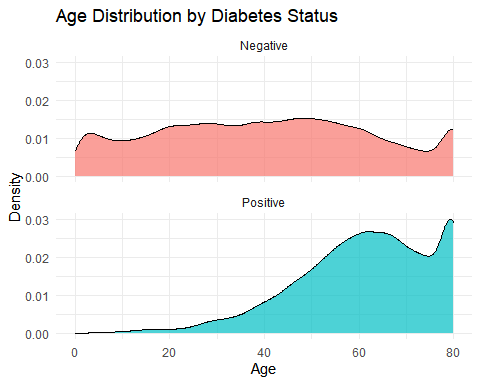
# Target distribution  
ggplot(df, aes(diabetes, fill = diabetes)) +  
 geom\_bar() +  
 geom\_text(  
 stat = "count",   
 aes(label = ..count..),   
 vjust = -0.5  
 ) +  
 labs(title = "Diabetes Class Distribution", x = "Diabetes Status")

## Warning: The dot-dot notation (`..count..`) was deprecated in ggplot2 3.4.0.  
## ℹ Please use `after\_stat(count)` instead.  
## This warning is displayed once every 8 hours.  
## Call `lifecycle::last\_lifecycle\_warnings()` to see where this warning was  
## generated.



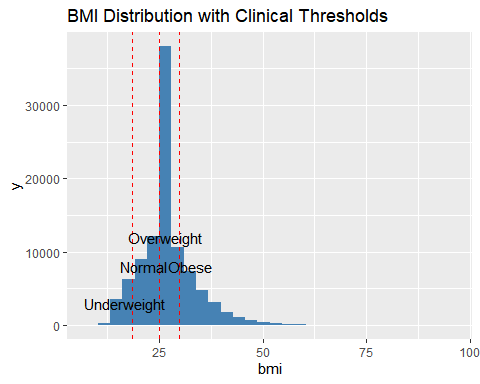
## Diabetes Versus Age Distribution

# Age analysis  
ggplot(df, aes(x = age, fill = diabetes, group = diabetes)) +  
 geom\_density(alpha = 0.7) +  
 facet\_wrap(~diabetes, ncol = 1) +  
 labs(title = "Age Distribution by Diabetes Status", x = "Age", y = "Density") +  
 theme\_minimal() +  
 theme(legend.position = "none")



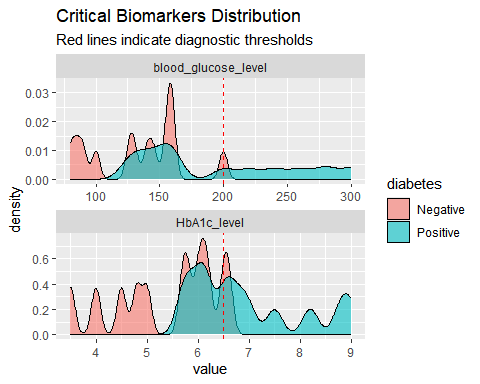
## Distribution of BMI

# BMI analysis with clinical thresholds  
ggplot(df, aes(bmi)) +  
 geom\_histogram(fill = "steelblue", bins = 30) +  
 geom\_vline(xintercept = c(18.5, 25, 30), linetype = "dashed", color = "red") +  
 annotate("text",   
 x = c(17, 21.5, 26.5, 32.5),   
 y = c(3000, 8000, 12000, 8000),   
 label = c("Underweight", "Normal", "Overweight", "Obese")) +  
 labs(title = "BMI Distribution with Clinical Thresholds")



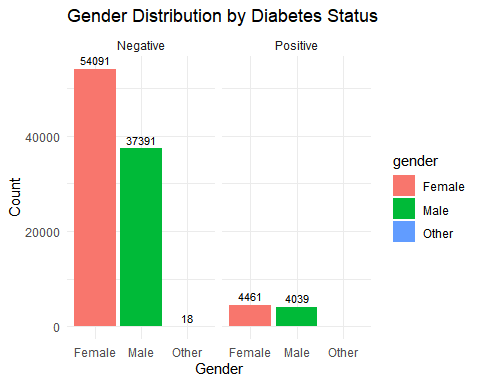
## Critical Biomarkers Versus Diabetes Class

df %>%   
 select(HbA1c\_level, blood\_glucose\_level, diabetes) %>%   
 pivot\_longer(-diabetes) %>%   
 ggplot(aes(x = value, fill = diabetes)) +  
 geom\_density(alpha = 0.6) +  
 facet\_wrap(~name, scales = "free", ncol = 1) +  
 geom\_vline(data = data.frame(name = c("HbA1c\_level", "blood\_glucose\_level"),  
 threshold = c(6.5, 200)),  
 aes(xintercept = threshold), color = "red", linetype = "dashed") +  
 labs(title = "Critical Biomarkers Distribution",  
 subtitle = "Red lines indicate diagnostic thresholds")



## Diabetes Class Distribution by Gender

ggplot(df, aes(x = gender, fill = gender)) +  
 geom\_bar() +  
 geom\_text(  
 stat = "count",  
 aes(label = after\_stat(count)),  
 vjust = -0.5,  
 size = 3  
 ) +  
 facet\_wrap(~diabetes) + # Add faceting by diabetes status  
 labs(  
 title = "Gender Distribution by Diabetes Status",  
 x = "Gender",  
 y = "Count"  
 ) +  
 theme\_minimal()



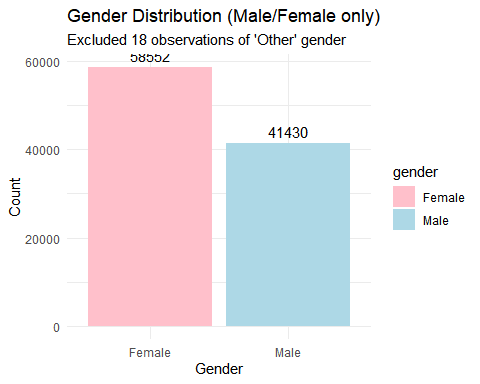
We are dropping “others” because they are only 18 observations (0.018% of 100k dataset) won’t impact analysis

## Distribution of Gender

table(df$gender)

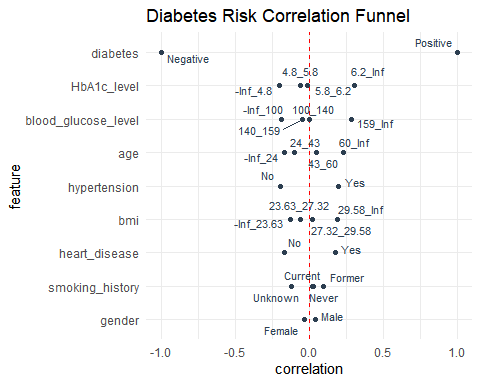
##   
## Female Male Other   
## 58552 41430 18

df <- df %>%  
 filter(gender %in% c("Male", "Female"))  
  
#Drop unused factor levels  
df$gender <- droplevels(df$gender)  
  
# Create the plot  
ggplot(df, aes(x = gender, fill = gender)) +  
 geom\_bar() +  
 geom\_text(  
 stat = "count",  
 aes(label = after\_stat(count)),  
 vjust = -0.5,  
 size = 4,  
 color = "black"  
 ) +  
 labs(  
 title = "Gender Distribution (Male/Female only)",  
 subtitle = "Excluded 18 observations of 'Other' gender",  
 x = "Gender",  
 y = "Count"  
 ) +  
 scale\_fill\_manual(values = c("Female" = "pink", "Male" = "lightblue")) +  
 theme\_minimal()



## Correlation Funnel for Target-Focused Relationships

# Correlation funnel for target-focused relationships  
  
# Prepare data  
df <- df%>%  
 mutate(  
 diabetes = factor(diabetes, levels = c("Negative", "Positive")),  
 across(where(is.character), as.factor)  
 )  
  
# Create correlation funnel  
df %>%  
 binarize() %>%  
 correlate(target = diabetes\_\_Positive) %>%   
 plot\_correlation\_funnel(interactive = FALSE) +  
 labs(title = "Diabetes Risk Correlation Funnel") +  
 theme\_minimal()



## Key Findings from Correlation Analysis

High-Risk Indicators • HbA1c > 6.2% & glucose > 140 mg/dL • Age > 60, hypertension, heart disease

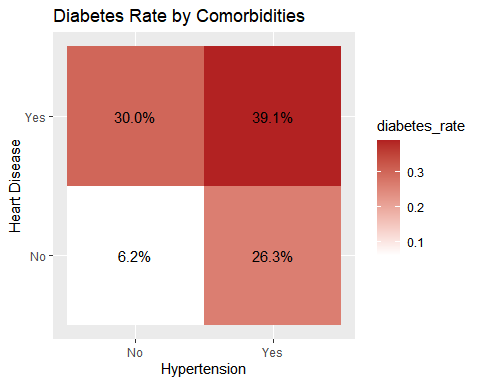
Protective Factors • HbA1c < 4.8% & glucose < 100 mg/dL • BMI < 23.63, age < 24

Notable Patterns

Clear biomarker thresholds Progressive age-risk relationship Gender shows minor association

## Diabetes Rate by Comorbidities

# Comorbidity analysis  
comorbidity\_analysis <- df %>%  
 group\_by(hypertension, heart\_disease) %>%  
 summarise(  
 diabetes\_rate = mean(diabetes == "Positive"),  
 count = n(),  
 .groups = "drop"  
 )  
  
ggplot(comorbidity\_analysis, aes(hypertension, heart\_disease, fill = diabetes\_rate)) +  
 geom\_tile() +  
 geom\_text(aes(label = scales::percent(diabetes\_rate, accuracy = 0.1))) +  
 scale\_fill\_gradient(low = "white", high = "firebrick") +  
 labs(title = "Diabetes Rate by Comorbidities",  
 x = "Hypertension", y = "Heart Disease")



## Data Preprocessing

# Handle BMI outliers using quantile capping  
summary(df$bmi)

## Min. 1st Qu. Median Mean 3rd Qu. Max.   
## 10.01 23.63 27.32 27.32 29.58 95.69

#calculate the 0.5% and 99.5% percentiles  
bmi\_caps <- quantile(df$bmi, probs = c(0.005, 0.995), na.rm = TRUE)  
print(bmi\_caps)

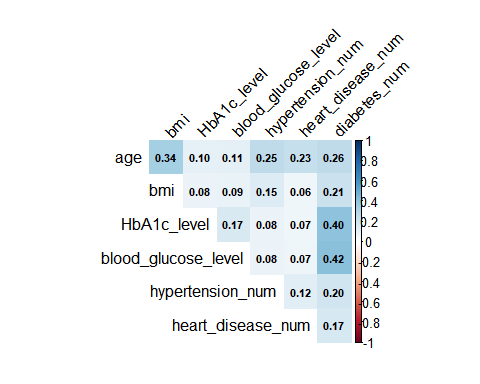
## 0.5% 99.5%   
## 13.89 52.50

The observed BMI values, ranging from 13.89 to 52.50, fall within the clinically plausible range of 12 to 60. This suggests that all recorded measurements are physiologically valid. As a result, we chose to retain all BMI values without applying any capping.

# Convert necessary variables to numeric for correlation  
corr\_data <- df %>%  
 mutate(  
 hypertension\_num = as.numeric(hypertension) - 1,   
 heart\_disease\_num = as.numeric(heart\_disease) - 1,  
 diabetes\_num = as.numeric(diabetes) - 1  
 ) %>%  
 select(age, bmi, HbA1c\_level, blood\_glucose\_level,   
 hypertension\_num, heart\_disease\_num, diabetes\_num)  
  
# Compute correlation matrix  
cor\_matrix <- cor(corr\_data)  
cor\_matrix

## age bmi HbA1c\_level blood\_glucose\_level  
## age 1.0000000 0.33737773 0.10134273 0.11067860  
## bmi 0.3373777 1.00000000 0.08301683 0.09131049  
## HbA1c\_level 0.1013427 0.08301683 1.00000000 0.16683088  
## blood\_glucose\_level 0.1106786 0.09131049 0.16683088 1.00000000  
## hypertension\_num 0.2511634 0.14767895 0.08094138 0.08443610  
## heart\_disease\_num 0.2333524 0.06120324 0.06759147 0.07007122  
## diabetes\_num 0.2579992 0.21437674 0.40069621 0.41958740  
## hypertension\_num heart\_disease\_num diabetes\_num  
## age 0.25116343 0.23335239 0.2579992  
## bmi 0.14767895 0.06120324 0.2143767  
## HbA1c\_level 0.08094138 0.06759147 0.4006962  
## blood\_glucose\_level 0.08443610 0.07007122 0.4195874  
## hypertension\_num 1.00000000 0.12125261 0.1978107  
## heart\_disease\_num 0.12125261 1.00000000 0.1717178  
## diabetes\_num 0.19781073 0.17171780 1.0000000

#correlation  
corrplot(cor\_matrix,   
 method = "color",   
 type = "upper",   
 tl.col = "black",   
 tl.srt = 45,   
 addCoef.col = "black",   
 number.cex = 0.7,   
 diag = FALSE)



# Print notable correlations (> 0.3)  
notable <- which(abs(cor\_matrix) > 0.3 & upper.tri(cor\_matrix), arr.ind = TRUE)  
  
if(nrow(notable) > 0) {  
 cat("\nNotable Correlations:\n")  
 for(i in 1:nrow(notable)) {  
 row <- notable[i,1]  
 col <- notable[i,2]  
 cat(rownames(cor\_matrix)[row], "&",   
 colnames(cor\_matrix)[col], ":",   
 round(cor\_matrix[row,col], 2), "\n")  
 }  
} else {  
 cat("\nNo notable correlations found (all r < 0.3)\n")  
}

##   
## Notable Correlations:  
## age & bmi : 0.34   
## HbA1c\_level & diabetes\_num : 0.4   
## blood\_glucose\_level & diabetes\_num : 0.42

The correlation matrix reveals moderate relationships between key clinical variables and diabestes, so we won’t change anything

# Create clinically relevant features  
df <- df %>%  
 mutate(  
 high\_hba1c = factor(ifelse(HbA1c\_level >= 6.5, "Yes", "No")),  
 high\_glucose = factor(ifelse(blood\_glucose\_level >= 200, "Yes", "No")),  
 age\_group = cut(age, breaks = c(0, 30, 45, 60, 100),  
 labels = c("<30", "30-45", "46-60", "60+")),  
 bmi\_category = cut(bmi, breaks = c(0, 18.5, 25, 30, 100),  
 labels = c("Underweight", "Normal", "Overweight", "Obese"))  
 )  
  
head(df)

## gender age hypertension heart\_disease smoking\_history bmi HbA1c\_level  
## 1 Female 80 No Yes Never 25.19 6.6  
## 2 Female 54 No No Unknown 27.32 6.6  
## 3 Male 28 No No Never 27.32 5.7  
## 4 Female 36 No No Current 23.45 5.0  
## 5 Male 76 Yes Yes Current 20.14 4.8  
## 6 Female 20 No No Never 27.32 6.6  
## blood\_glucose\_level diabetes high\_hba1c high\_glucose age\_group bmi\_category  
## 1 140 Negative Yes No 60+ Overweight  
## 2 80 Negative Yes No 46-60 Overweight  
## 3 158 Negative No No <30 Overweight  
## 4 155 Negative No No 30-45 Normal  
## 5 155 Negative No No 60+ Normal  
## 6 85 Negative Yes No <30 Overweight

# prepare modeling data  
model\_data <- df %>%  
 select(-age\_group, -bmi\_category) %>%  
 mutate(diabetes = as.factor(diabetes))  
  
# create encoding model (EXCLUDING diabetes)  
dummy\_model <- dummyVars(~ . ,   
 data = model\_data %>% select(-diabetes),  
 fullRank = TRUE)  
  
# apply encoding to predictors only  
encoded\_predictors <- predict(dummy\_model, newdata = model\_data) %>%   
 as.data.frame()  
  
# add target variable back  
final\_data <- encoded\_predictors %>%   
 mutate(diabetes = model\_data$diabetes)

Test/Train/Validation Splits

set.seed(35)  
split\_df <- final\_data  
  
train\_index <- createDataPartition(final\_data$diabetes, p = 0.8, list = FALSE)  
train\_data <- final\_data[train\_index, ]  
test\_val\_data <- final\_data[-train\_index, ]  
  
test\_val\_index <- createDataPartition(test\_val\_data$diabetes, p = 0.5, list = FALSE)  
validation\_data <- test\_val\_data[test\_val\_index, ]  
test\_data <- test\_val\_data[-test\_val\_index, ]  
  
dim(final\_data)

## [1] 99982 13

dim(train\_data)

## [1] 79986 13

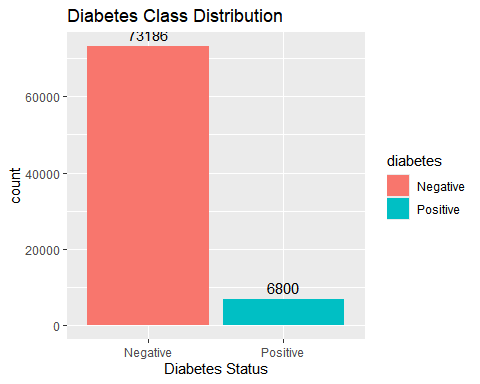
dim(test\_data)

## [1] 9998 13

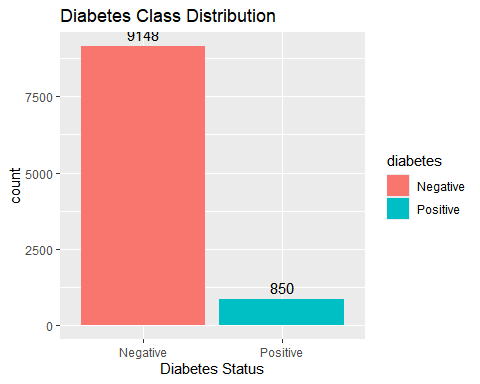
dim(validation\_data)

## [1] 9998 13

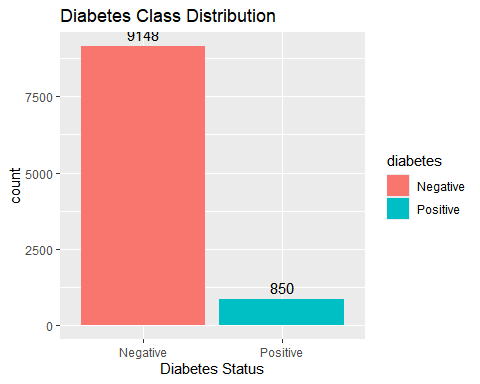
ggplot(train\_data, aes(diabetes, fill = diabetes)) +  
 geom\_bar() +  
 geom\_text(  
 stat = "count",   
 aes(label = ..count..),   
 vjust = -0.5  
 ) +  
 labs(title = "Diabetes Class Distribution", x = "Diabetes Status")



ggplot(test\_data, aes(diabetes, fill = diabetes)) +  
 geom\_bar() +  
 geom\_text(  
 stat = "count",   
 aes(label = ..count..),   
 vjust = -0.5  
 ) +  
 labs(title = "Diabetes Class Distribution", x = "Diabetes Status")



ggplot(validation\_data, aes(diabetes, fill = diabetes)) +  
 geom\_bar() +  
 geom\_text(  
 stat = "count",   
 aes(label = ..count..),   
 vjust = -0.5  
 ) +  
 labs(title = "Diabetes Class Distribution", x = "Diabetes Status")



# Model Development

## Niat’s Model - Random Forest

### Model Building

# Convert 'diabetes' to factor  
train\_data$diabetes <- as.factor(train\_data$diabetes)  
test\_data$diabetes <- as.factor(test\_data$diabetes)  
validation\_data$diabetes <- as.factor(validation\_data$diabetes)

set.seed(123)   
base\_rf <- randomForest(  
 diabetes ~ .,   
 data = train\_data,  
 importance = TRUE   
)  
  
validation\_pred\_base <- predict(base\_rf, validation\_data)  
confusionMatrix(validation\_pred\_base, validation\_data$diabetes)

## Confusion Matrix and Statistics  
##   
## Reference  
## Prediction Negative Positive  
## Negative 9142 287  
## Positive 6 563  
##   
## Accuracy : 0.9707   
## 95% CI : (0.9672, 0.9739)  
## No Information Rate : 0.915   
## P-Value [Acc > NIR] : < 2.2e-16   
##   
## Kappa : 0.7784   
##   
## Mcnemar's Test P-Value : < 2.2e-16   
##   
## Sensitivity : 0.9993   
## Specificity : 0.6624   
## Pos Pred Value : 0.9696   
## Neg Pred Value : 0.9895   
## Prevalence : 0.9150   
## Detection Rate : 0.9144   
## Detection Prevalence : 0.9431   
## Balanced Accuracy : 0.8308   
##   
## 'Positive' Class : Negative   
##

# Set up parallel processing  
cl <- makePSOCKcluster(4)   
registerDoParallel(cl)  
ctrl <- trainControl(  
 method = "cv",  
 number = 5,  
 classProbs = TRUE,  
 summaryFunction = twoClassSummary,  
 savePredictions = "final",  
 allowParallel = TRUE  
)   
  
  
tune\_grid <- expand.grid(  
 mtry = floor(seq(2, ncol(train\_data)-1, length.out = 5)),  
 splitrule = "gini",  
 min.node.size = 1  
)  
  
# Train model THIRD  
set.seed(123)  
tuned\_rf <- train(  
 diabetes ~ .,  
 data = train\_data,  
 method = "ranger",  
 trControl = ctrl,  
 tuneGrid = tune\_grid,  
 importance = "impurity",  
 num.trees = 100,  
 metric = "ROC"  
)  
  
# Stop parallel cluster  
stopCluster(cl)  
print(tuned\_rf$bestTune)

## mtry splitrule min.node.size  
## 2 4 gini 1

validation\_pred\_tuned <- predict(tuned\_rf, validation\_data)  
confusionMatrix(validation\_pred\_tuned, validation\_data$diabetes)

## Confusion Matrix and Statistics  
##   
## Reference  
## Prediction Negative Positive  
## Negative 9129 282  
## Positive 19 568  
##   
## Accuracy : 0.9699   
## 95% CI : (0.9664, 0.9732)  
## No Information Rate : 0.915   
## P-Value [Acc > NIR] : < 2.2e-16   
##   
## Kappa : 0.7749   
##   
## Mcnemar's Test P-Value : < 2.2e-16   
##   
## Sensitivity : 0.9979   
## Specificity : 0.6682   
## Pos Pred Value : 0.9700   
## Neg Pred Value : 0.9676   
## Prevalence : 0.9150   
## Detection Rate : 0.9131   
## Detection Prevalence : 0.9413   
## Balanced Accuracy : 0.8331   
##   
## 'Positive' Class : Negative   
##

### Model Testing and Evaluation

# function for bootstrap CIs  
get\_bootstrap\_ci <- function(predictions, true\_labels, metric = "accuracy", n\_bootstrap = 100) {  
 set.seed(123)  
 boot\_results <- replicate(n\_bootstrap, {  
 idx <- sample(length(true\_labels), replace = TRUE)  
 if (metric == "accuracy") {  
 mean(predictions[idx] == true\_labels[idx])  
 } else if (metric == "sensitivity") {  
 caret::sensitivity(predictions[idx], true\_labels[idx])  
 } else if (metric == "specificity") {  
 caret::specificity(predictions[idx], true\_labels[idx])  
 }  
 })  
 quantile(boot\_results, c(0.025, 0.975)) # 95% CI  
}  
  
cat("\n\*\*Base Model Performance\*\*\n")

##   
## \*\*Base Model Performance\*\*

confusionMatrix(validation\_pred\_base, validation\_data$diabetes)

## Confusion Matrix and Statistics  
##   
## Reference  
## Prediction Negative Positive  
## Negative 9142 287  
## Positive 6 563  
##   
## Accuracy : 0.9707   
## 95% CI : (0.9672, 0.9739)  
## No Information Rate : 0.915   
## P-Value [Acc > NIR] : < 2.2e-16   
##   
## Kappa : 0.7784   
##   
## Mcnemar's Test P-Value : < 2.2e-16   
##   
## Sensitivity : 0.9993   
## Specificity : 0.6624   
## Pos Pred Value : 0.9696   
## Neg Pred Value : 0.9895   
## Prevalence : 0.9150   
## Detection Rate : 0.9144   
## Detection Prevalence : 0.9431   
## Balanced Accuracy : 0.8308   
##   
## 'Positive' Class : Negative   
##

# accuracy CI  
ci\_accuracy\_base <- get\_bootstrap\_ci(validation\_pred\_base, validation\_data$diabetes, "accuracy")  
cat("\nBase Model Accuracy 95% CI:", ci\_accuracy\_base, "\n")

##   
## Base Model Accuracy 95% CI: 0.9672409 0.9743474

# tuned model metrics  
cat("\n\*\*Tuned Model Performance\*\*\n")

##   
## \*\*Tuned Model Performance\*\*

confusionMatrix(validation\_pred\_tuned, validation\_data$diabetes)

## Confusion Matrix and Statistics  
##   
## Reference  
## Prediction Negative Positive  
## Negative 9129 282  
## Positive 19 568  
##   
## Accuracy : 0.9699   
## 95% CI : (0.9664, 0.9732)  
## No Information Rate : 0.915   
## P-Value [Acc > NIR] : < 2.2e-16   
##   
## Kappa : 0.7749   
##   
## Mcnemar's Test P-Value : < 2.2e-16   
##   
## Sensitivity : 0.9979   
## Specificity : 0.6682   
## Pos Pred Value : 0.9700   
## Neg Pred Value : 0.9676   
## Prevalence : 0.9150   
## Detection Rate : 0.9131   
## Detection Prevalence : 0.9413   
## Balanced Accuracy : 0.8331   
##   
## 'Positive' Class : Negative   
##

# accuracy CI  
ci\_accuracy\_tuned <- get\_bootstrap\_ci(validation\_pred\_tuned, validation\_data$diabetes, "accuracy")  
cat("\nTuned Model Accuracy 95% CI:", ci\_accuracy\_tuned, "\n")

##   
## Tuned Model Accuracy 95% CI: 0.9662307 0.9733522

#AUC-ROC Comparison   
pred\_probs\_base <- predict(base\_rf, validation\_data, type = "prob")[, "Positive"]  
pred\_probs\_tuned <- predict(tuned\_rf, validation\_data, type = "prob")[, "Positive"]  
  
roc\_base <- roc(validation\_data$diabetes, pred\_probs\_base)

## Setting levels: control = Negative, case = Positive

## Setting direction: controls < cases

roc\_tuned <- roc(validation\_data$diabetes, pred\_probs\_tuned)

## Setting levels: control = Negative, case = Positive  
## Setting direction: controls < cases

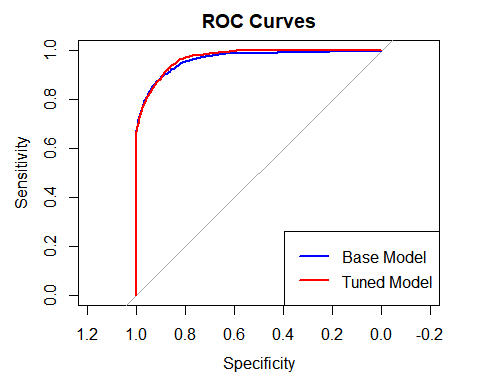
cat("\nBase Model AUC:", auc(roc\_base), "95% CI:", ci.auc(roc\_base), "\n")

##   
## Base Model AUC: 0.96451 95% CI: 0.9579397 0.96451 0.9710803

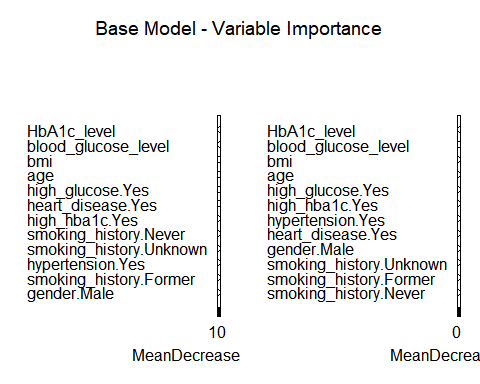
cat("Tuned Model AUC:", auc(roc\_tuned), "95% CI:", ci.auc(roc\_tuned), "\n")

## Tuned Model AUC: 0.9704229 95% CI: 0.9657961 0.9704229 0.9750498

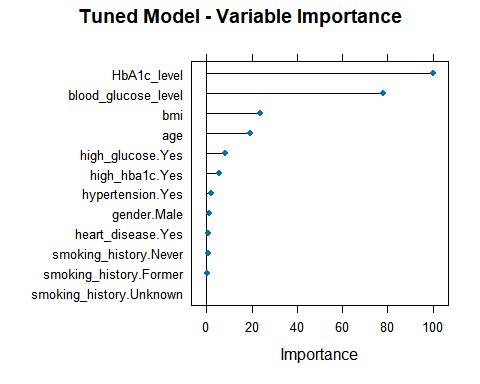
# Plot ROC curves  
plot(roc\_base, col = "blue", main = "ROC Curves")  
lines(roc\_tuned, col = "red")  
legend("bottomright", legend = c("Base Model", "Tuned Model"), col = c("blue", "red"), lwd = 2)



# base model  
varImpPlot(base\_rf, main = "Base Model - Variable Importance")



# tuned model  
plot(varImp(tuned\_rf), main = "Tuned Model - Variable Importance")



## Tommy’s Model - SVM

### Model Building

ctrl <- trainControl(  
 method = "cv",  
 number = 5,  
 classProbs = TRUE,  
 summaryFunction = twoClassSummary  
)  
  
cost\_varies <- c(0.01, 0.1, 1, 10)  
eps\_varies <- c(0.01, 0.1, 0.5, 1.0)  
sig\_varies <- c(0.01, 0.1, 1, 10)  
  
svm\_models\_cost <- list()  
svm\_models\_eps <- list()  
svm\_models\_sig <- list()  
  
for (C\_val in cost\_varies) {  
 svm\_models\_cost[[as.character(C\_val)]] <- ksvm(  
 diabetes ~.,  
 data = train\_data,  
 kernel = "rbfdot",  
 trControl = ctrl,  
 prob.model = TRUE,  
 kpar = list(sigma = 1),  
 C = C\_val,  
 epsilon = 1  
 )  
}

## line search fails -15.9598 -12.47955 2.120423e-05 -1.724193e-05 -1.846131e-07 -1.69242e-07 -9.965208e-13

for (E\_val in eps\_varies) {  
 svm\_models\_eps[[as.character(E\_val)]] <- ksvm(  
 diabetes ~.,  
 data = train\_data,  
 kernel = "rbfdot",  
 trControl = ctrl,  
 prob.model = TRUE,  
 kpar = list(sigma = 1),  
 C = 1,  
 epsilon = E\_val  
 )  
}  
  
for (S\_val in sig\_varies) {  
 svm\_models\_sig[[as.character(S\_val)]] <- ksvm(  
 diabetes ~.,  
 data = train\_data,  
 kernel = "rbfdot",  
 trControl = ctrl,  
 prob.model = TRUE,  
 kpar = list(sigma = S\_val),  
 C = 1,  
 epsilon = 1  
 )  
}

## line search fails -7.752801 -4.131246 1.069888e-05 -9.148427e-06 -2.377054e-08 -1.678207e-08 -1.007887e-13

for(C\_val in names(svm\_models\_cost)) {  
 cost\_model <- svm\_models\_cost[[C\_val]]  
 predictions <- predict(cost\_model, validation\_data)  
 cat("Confusion Matrix for C =", C\_val, "\n")  
 print(table(Predicted = predictions, Actual = test\_data$diabetes))  
 cat("\n")  
}

## Confusion Matrix for C = 0.01   
## Actual  
## Predicted Negative Positive  
## Negative 9148 850  
## Positive 0 0  
##   
## Confusion Matrix for C = 0.1   
## Actual  
## Predicted Negative Positive  
## Negative 8892 835  
## Positive 256 15  
##   
## Confusion Matrix for C = 1   
## Actual  
## Predicted Negative Positive  
## Negative 8659 808  
## Positive 489 42  
##   
## Confusion Matrix for C = 10   
## Actual  
## Predicted Negative Positive  
## Negative 8589 803  
## Positive 559 47

for(E\_val in names(svm\_models\_eps)) {  
 eps\_model <- svm\_models\_eps[[E\_val]]  
 predictions <- predict(eps\_model, validation\_data)  
 cat("Confusion Matrix for epsilon =", E\_val, "\n")  
 print(table(Predicted = predictions, Actual = test\_data$diabetes))  
 cat("\n")  
}

## Confusion Matrix for epsilon = 0.01   
## Actual  
## Predicted Negative Positive  
## Negative 8659 808  
## Positive 489 42  
##   
## Confusion Matrix for epsilon = 0.1   
## Actual  
## Predicted Negative Positive  
## Negative 8659 808  
## Positive 489 42  
##   
## Confusion Matrix for epsilon = 0.5   
## Actual  
## Predicted Negative Positive  
## Negative 8659 808  
## Positive 489 42  
##   
## Confusion Matrix for epsilon = 1   
## Actual  
## Predicted Negative Positive  
## Negative 8659 808  
## Positive 489 42

for(S\_val in names(svm\_models\_sig)) {  
 sig\_model <- svm\_models\_sig[[S\_val]]  
 predictions <- predict(sig\_model, validation\_data)  
 cat("Confusion Matrix for sigma =", S\_val, "\n")  
 print(table(Predicted = predictions, Actual = test\_data$diabetes))  
 cat("\n")  
}

## Confusion Matrix for sigma = 0.01   
## Actual  
## Predicted Negative Positive  
## Negative 8723 814  
## Positive 425 36  
##   
## Confusion Matrix for sigma = 0.1   
## Actual  
## Predicted Negative Positive  
## Negative 8662 811  
## Positive 486 39  
##   
## Confusion Matrix for sigma = 1   
## Actual  
## Predicted Negative Positive  
## Negative 8659 808  
## Positive 489 42  
##   
## Confusion Matrix for sigma = 10   
## Actual  
## Predicted Negative Positive  
## Negative 8996 836  
## Positive 152 14

svm\_model\_base <- ksvm(  
 diabetes ~.,  
 data = train\_data,  
 kernel = "rbfdot",  
 trControl = ctrl,  
 prob.model = TRUE,  
 kpar = list(sigma = 1),  
 C = 1,  
 epsilon = 1  
)  
  
svm\_model\_tuned <- ksvm(  
 diabetes ~.,  
 data = train\_data,  
 kernel = "rbfdot",  
 trControl = ctrl,  
 prob.model = TRUE,  
 kpar = list(sigma = 0.1),  
 C = 10,  
 epsilon = 1  
)

### Model Testing and Evaluation

basePredictions <- predict(svm\_model\_base, test\_data)  
cat("Confusion Matrix for Base SVM Model", "\n")

## Confusion Matrix for Base SVM Model

print(table(Predicted = basePredictions, Actual = test\_data$diabetes))

## Actual  
## Predicted Negative Positive  
## Negative 9128 315  
## Positive 20 535

cat("\n")

tunedPredictions <- predict(svm\_model\_tuned, test\_data)  
cat("Confusion Matrix for Tuned SVM Model", "\n")

## Confusion Matrix for Tuned SVM Model

print(table(Predicted = tunedPredictions, Actual = test\_data$diabetes))

## Actual  
## Predicted Negative Positive  
## Negative 9141 273  
## Positive 7 577

cat("\n")

accuracy\_function <- function(data, indices) {  
 d <- data[indices, ]  
 mean(d$actual == d$predicted)  
}  
  
dfBase <- data.frame(  
 actual = test\_data$diabetes,  
 predicted = basePredictions  
)  
  
boot\_out\_base <- boot(data = dfBase, statistic = accuracy\_function, R = 1000)  
  
cat("Confidence Interval for Base SVM Model", "\n")

## Confidence Interval for Base SVM Model

boot.ci(boot\_out\_base, type = "perc")

## BOOTSTRAP CONFIDENCE INTERVAL CALCULATIONS  
## Based on 1000 bootstrap replicates  
##   
## CALL :   
## boot.ci(boot.out = boot\_out\_base, type = "perc")  
##   
## Intervals :   
## Level Percentile   
## 95% ( 0.9630, 0.9698 )   
## Calculations and Intervals on Original Scale

cat("\n")

dfTuned <- data.frame(  
 actual = test\_data$diabetes,  
 predicted = tunedPredictions  
)  
  
boot\_out\_tuned <- boot(data = dfTuned, statistic = accuracy\_function, R = 1000)  
  
cat("Confidence Interval for Tuned SVM Model", "\n")

## Confidence Interval for Tuned SVM Model

boot.ci(boot\_out\_tuned, type = "perc")

## BOOTSTRAP CONFIDENCE INTERVAL CALCULATIONS  
## Based on 1000 bootstrap replicates  
##   
## CALL :   
## boot.ci(boot.out = boot\_out\_tuned, type = "perc")  
##   
## Intervals :   
## Level Percentile   
## 95% ( 0.9688, 0.9754 )   
## Calculations and Intervals on Original Scale

cat("\n")

basePredProb <- predict(svm\_model\_base, test\_data, type = "prob")[, "Positive"]  
tunedPredProb <- predict(svm\_model\_tuned, test\_data, type = "prob")[, "Positive"]  
  
baseSVMROC <- roc(test\_data$diabetes, basePredProb)

## Setting levels: control = Negative, case = Positive

## Setting direction: controls < cases

tunedSVMROC <- roc(test\_data$diabetes, tunedPredProb)

## Setting levels: control = Negative, case = Positive  
## Setting direction: controls < cases

cat("\nBase SVM Model AUC:", auc(baseSVMROC), "95% CI:", ci.auc(baseSVMROC))

##   
## Base SVM Model AUC: 0.9216419 95% CI: 0.9079377 0.9216419 0.9353461

cat("\nTuned SVM Model AUC:", auc(tunedSVMROC), "95% CI:", ci.auc(tunedSVMROC))

##   
## Tuned SVM Model AUC: 0.9266505 95% CI: 0.9140109 0.9266505 0.9392901

plot(baseSVMROC, col = "blue", main = "ROC Curves")  
lines(tunedSVMROC, col = "red")

