#step1

import warnings

warnings.filterwarnings('ignore')

import pandas as pd

import numpy as np

from matplotlib import pyplot as plt

import seaborn as sns

from sklearn.model\_selection import train\_test\_split

from sklearn.linear\_model import LogisticRegression

from sklearn.metrics import confusion\_matrix, classification\_report

from sklearn.preprocessing import StandardScaler, MinMaxScaler

from sklearn.metrics import accuracy\_score, classification\_report, confusion\_matrix

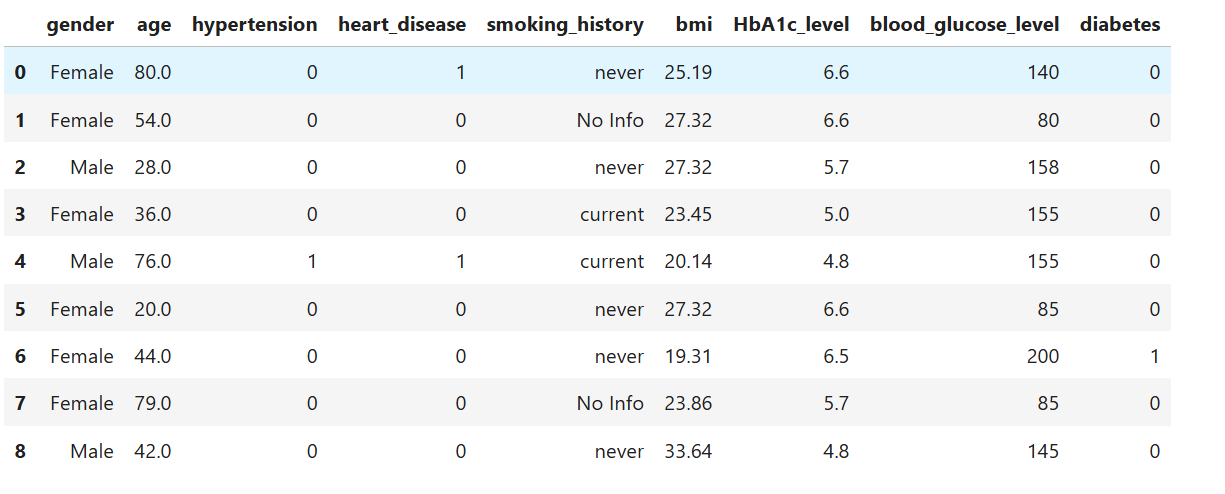
from sklearn.ensemble import RandomForestClassifier

#step2

#importing dataset

df = pd.read\_csv("D:\\diabetes\_prediction\_dataset.csv")

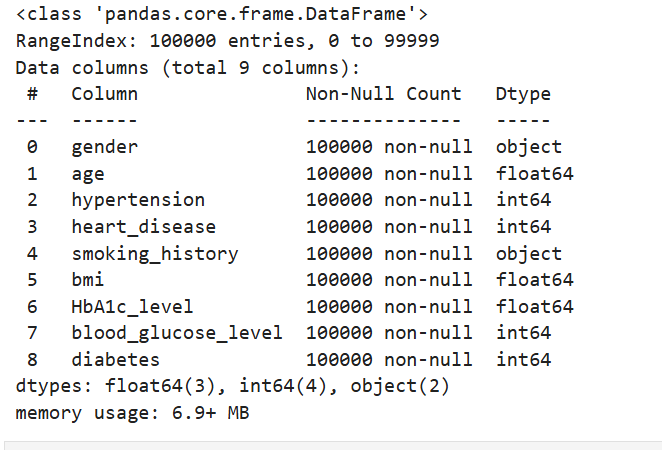
df.head(9)



#step3

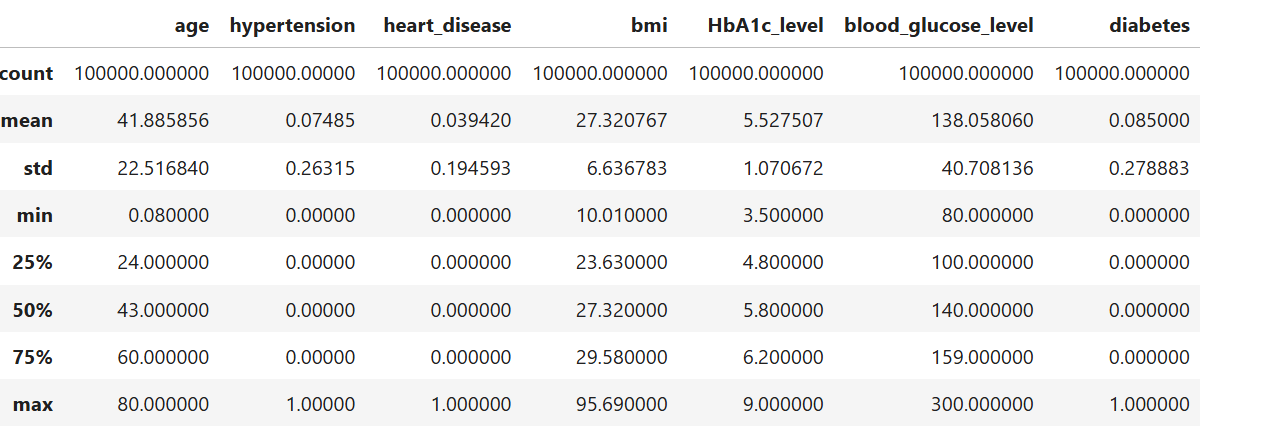
#getting info about all the columns

df.info()

  
#step4

#getting statiscal data about each numerical columns

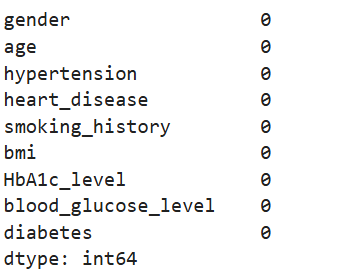
df.describe()



#Step5

#checking if dataset contains any null values

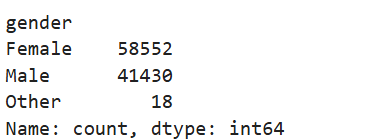
df.isna().sum()



#step6

#checking number of unique values in a column

df['gender'].value\_counts()



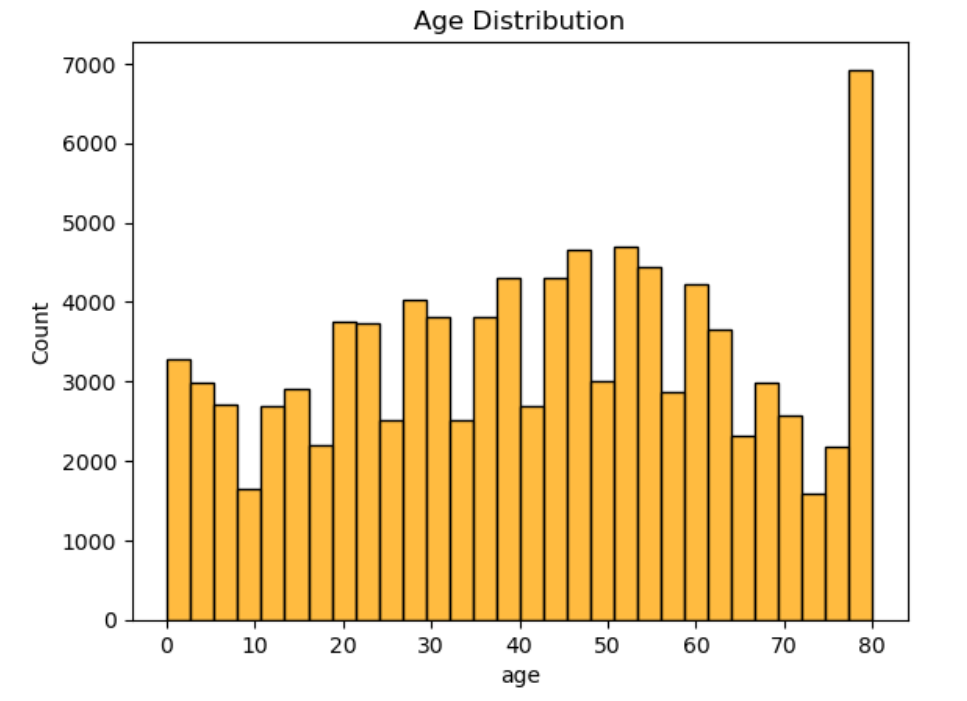
#step7

#Age Distribution

sns.histplot(df['age'], bins=30, color='orange')

plt.title("Age Distribution")

plt.show()

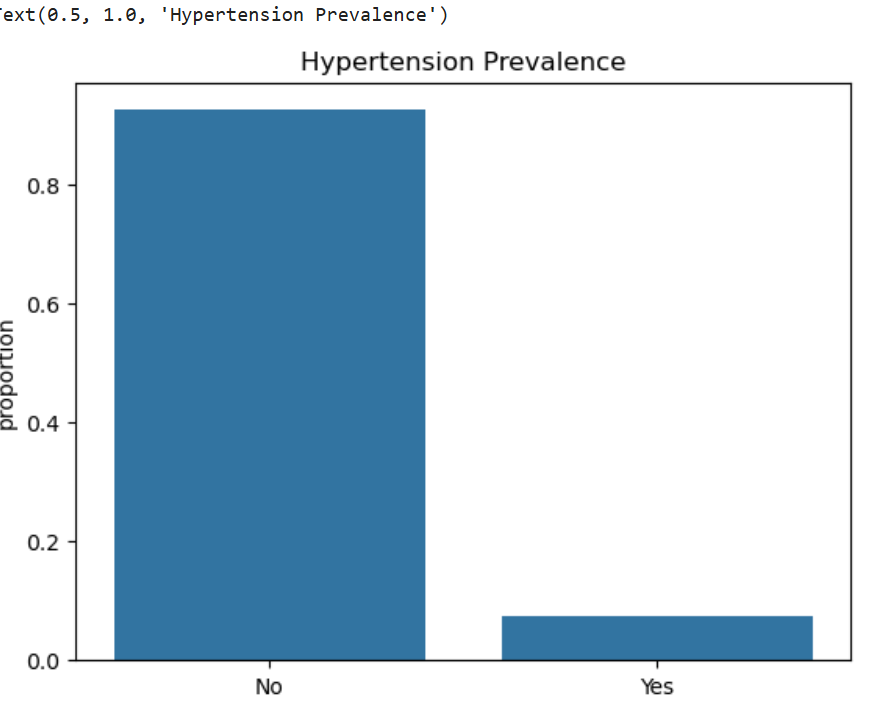


#step8

#Hypertension & Heart Disease Prevalence

sns.barplot(x=['No', 'Yes'], y=df['hypertension'].value\_counts(normalize=True))

plt.title("Hypertension Prevalence")

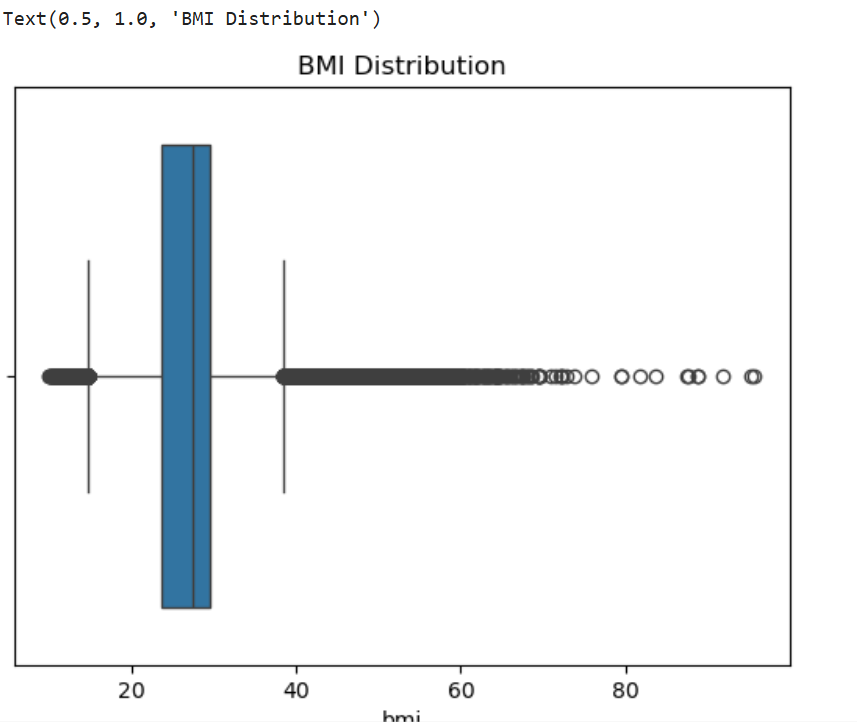


#step9

#BMI Distribution ans Outliers

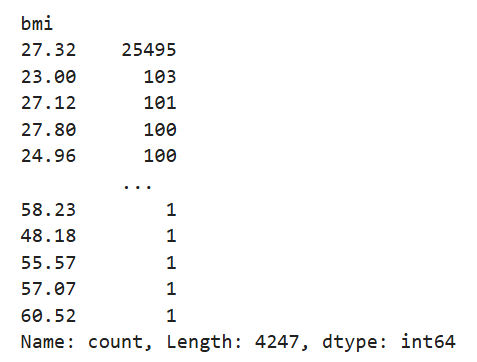
sns.boxplot(x=df['bmi'])

plt.title("BMI Distribution")



#step10

df['bmi'].value\_counts()



#step11

#Performing outlier treatment on BMI

Q1 = df['bmi'].quantile(0.25)

Q3 = df['bmi'].quantile(0.75)

IQR = Q3 - Q1

lower\_bound = Q1 - 1.5 \* IQR

upper\_bound = Q3 + 1.5 \* IQR

# Finding outliers

outliers = df[(df['bmi'] < lower\_bound) | (df['bmi'] > upper\_bound)]

print(f'Number of outliers in BMI column: {len(outliers)}')

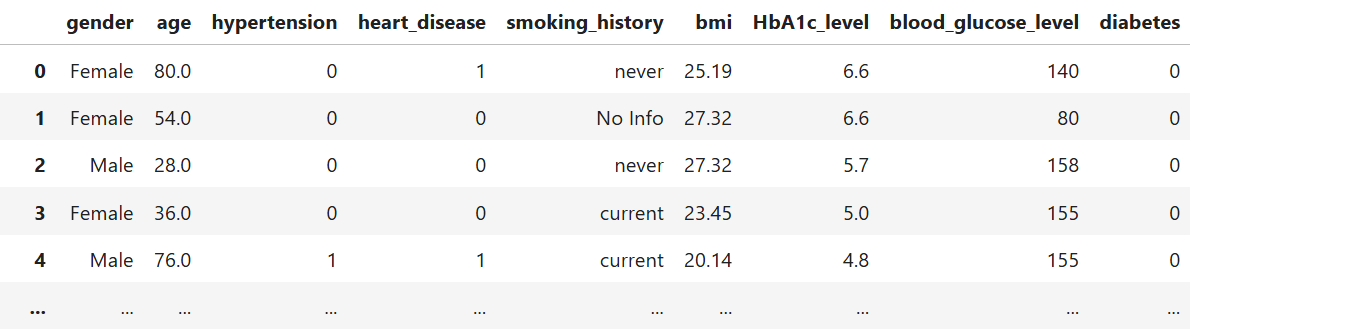


#step12

#New df with no outliers in BMI column

df = df[(df['bmi'] >= lower\_bound) & (df['bmi'] <= upper\_bound)]

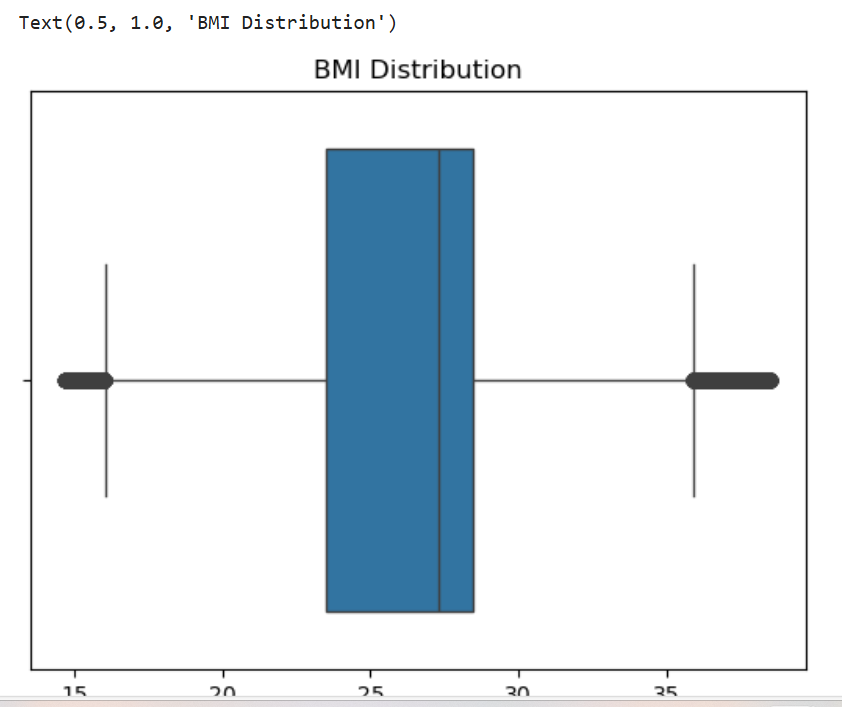
df



#step13

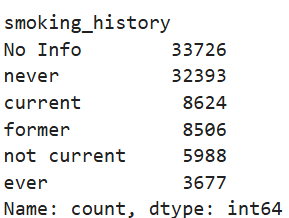
sns.boxplot(x=df['bmi'])

plt.title("BMI Distribution")



#step14

df.smoking\_history.value\_counts()



#step15

#Based on smoking , mapping to a value

smoking\_history\_map = {

'never': 0, # Non-Smoker

'not current': 0, # Non-Smoker

'former': 1, # Former Smoker

'current': 2, # Smoker

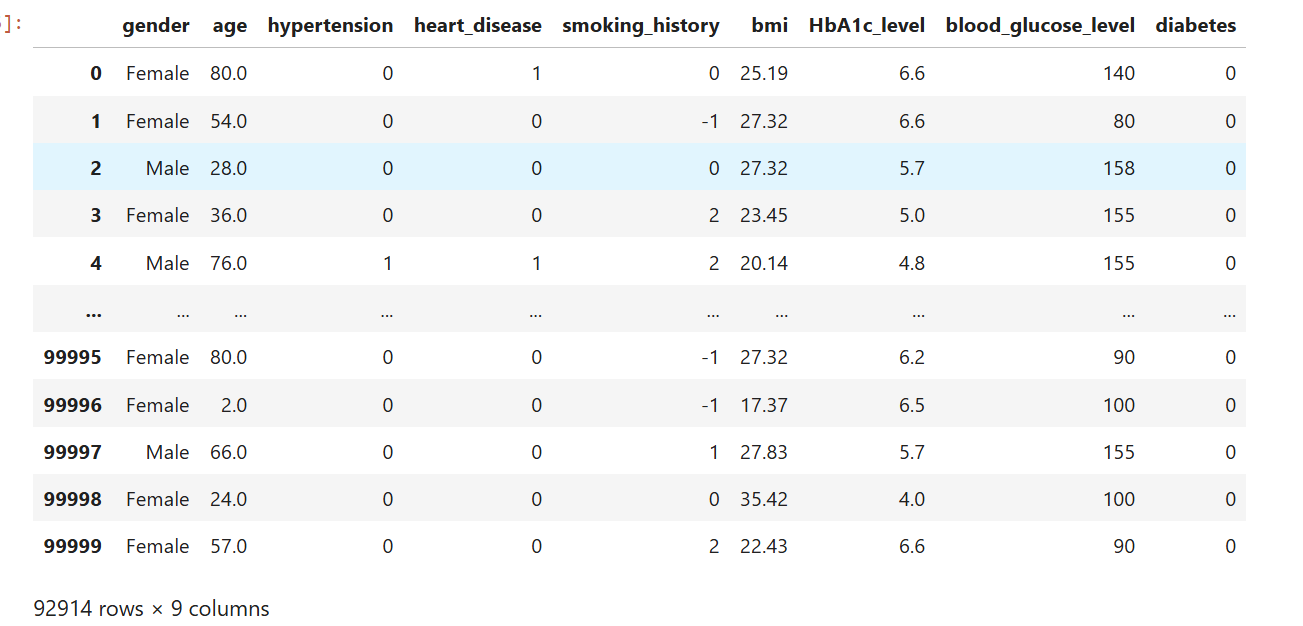
'ever': 2, # Smoker (considered similar to 'current')

'No Info': -1 # Unknown

}

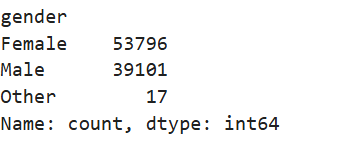
df['smoking\_history'] = df['smoking\_history'].map(smoking\_history\_map)

df



#step16

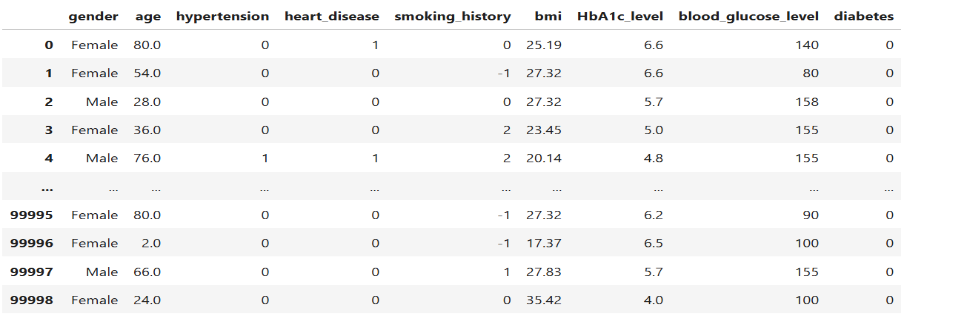
df['gender'].value\_counts()

   
#step17

# Remove rows where gender is "Other"

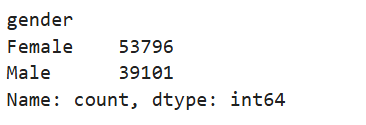
df = df[df['gender'] != 'Other']

df



#step18

df['gender'].value\_counts()



#step19

# Mapping dictionary for gender (Female: 0, Male: 1)

gender\_map = {

'Female': 0,

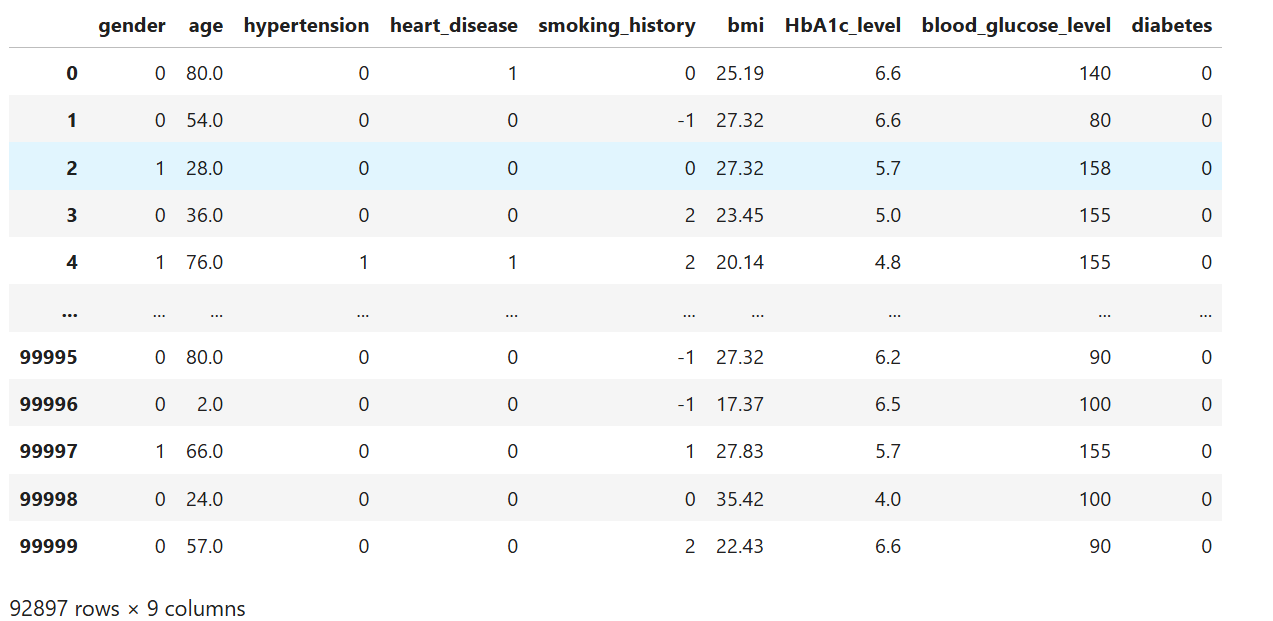
'Male': 1

}

# Apply the mapping to the 'gender' column

df['gender'] = df['gender'].map(gender\_map)

df



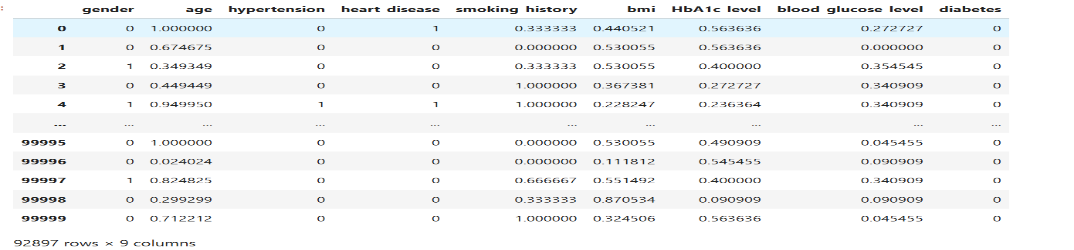
#Step20  
#scaling data to bring all values to a Range [0,1]

scaler = MinMaxScaler()

# Apply scaling to the columns (exclude categorical ones)

df[['age','smoking\_history','bmi','HbA1c\_level', 'blood\_glucose\_level']] = scaler.fit\_transform(df[['age','smoking\_history','bmi','HbA1c\_level', 'blood\_glucose\_level']])

df



#step21

X = df.drop('diabetes', axis=1)

y = df['diabetes']

# Train-test split

X\_train, X\_test, y\_train, y\_test = train\_test\_split(X, y, test\_size=0.3, random\_state=42)

#step22

# Initialize the Logistic Regression Model

model\_logistic = LogisticRegression(class\_weight='balanced', random\_state=42)

# Train the model

model\_logistic.fit(X\_train, y\_train)

# Make predictions

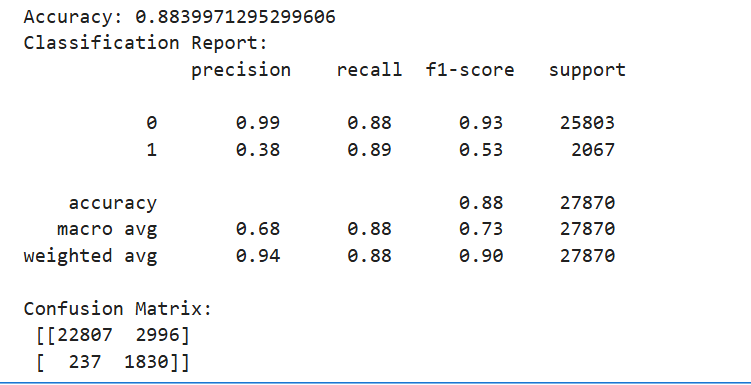
y\_pred = model\_logistic.predict(X\_test)

# Evaluate the model

print("Accuracy:", accuracy\_score(y\_test, y\_pred))

print("Classification Report:\n", classification\_report(y\_test, y\_pred))

print("Confusion Matrix:\n", confusion\_matrix(y\_test, y\_pred))



#step23

# Initialize the Random Forest Classifier model

model\_rfc = RandomForestClassifier(class\_weight='balanced', random\_state=42)

# Train the model

model\_rfc.fit(X\_train, y\_train)

# Make predictions

y\_pred = model\_rfc.predict(X\_test)

# Evaluate the model

print("Accuracy:", accuracy\_score(y\_test, y\_pred))

print("Classification Report:\n", classification\_report(y\_test, y\_pred))

print("Confusion Matrix:\n", confusion\_matrix(y\_test, y\_pred))

