PREDICTIVE ANALYSIS OF DIABETES STATUS USING MACHINE LEARNING MODEL

SHARMILASHREE V, PAVITHRA KADRI, DEEPTHI MALLEPALLY, AMULYA RAGULA, DEVAHARSHAGUBBALA

Indiana University-Purdue University, Indianapolis, IN 46202, USA,
Vsha@iu.edu, pkadri@iu.edu, dmallepa@iu.edu, aragula@iu.edu , Degubb@iu.edu

ABSTRACT:

The study explores the use of machine learning models for predicting diabetes status, which is a significant public health concern due to its increasing prevalence globally. Using a comprehensive dataset from Iraqi healthcare centers, our team focused on categorizing individuals as diabetic, pre-diabetic, or non-diabetic based on numerous health parameters. The project included comprehensive data analysis, statistical testing, feature engineering, and evaluation of various machine learning classifiers. The models demonstrated notable accuracy and precision in diabetes prediction, highlighting the potential of machine learning in assisting healthcare professionals and addressing the diabetes epidemic. This study contributes towards improving early detection and personalized treatment of diabetes, eventually aiming to improve patient outcomes and public health

KEYWORDS:

Diabetes, Diabetes Classification, Diabetes Risk Factors, Machine Learning, Health Parameters, HbA1c, Early Detection, Risk Factor Analysis, Predictive Modeling, Public Health, Patient Outcomes, Python

1 PROJECT SCOPE

1.1 INTRODUCTION:

Diabetes, a significant global public health concern, is a chronic metabolic disorder characterized by elevated blood sugar levels, leading to serious health complications like cardiovascular disease and kidney failure. (American Diabetes Association, 2021) highlights the complexity and severity of diabetes as a health issue. As per the International Diabetes Federation Diabetes Atlas (9th Edition), the prevalence of diabetes has been rapidly increasing, with 463 million individuals affected in 2019, and projections suggesting a rise to 700 million by 2045 (Saeedi et al., 2019). This trend emphasizes the critical need for improved diagnostic and management strategies for diabetes.

Our project uses a retrospective observational method and explores the usage of machine learning models for the predictive analysis of diabetes status. Using a dataset generated from the Medical City Hospital and the Specialized Center for Endocrinology and Diabetes at Al-Kindy Teaching Hospital in Iraq (Rashid, 2020), which includes numerous medical parameters such as age, gender, BMI, blood glucose,

cholesterol, and HbA1c.We aim to enhance diabetes detection and treatment while also addressing its impact on individuals and society.

1.2 AIM:

The aim of the project is to build machine learning models that can accurately classify patients as diabetic, pre-diabetic, or non-diabetic based on their health parameters. These models will aid healthcare professionals in identifying individuals who are at risk of developing diabetes, resulting in early detection and timely intervention to address this aim we posted research question:

- Do factors such as age, urea, HbA1c, cholesterol, VLDL, BMI, and gender have a statistically significant correlation with the classification of individuals as diabetic, non-diabetic, or pre-diabetic?

The following two hypotheses are based on the research question:

- **Null Hypothesis (H0):** There is no statistically significant correlation between factors such as age, urea, HbA1c, cholesterol, VLDL, BMI, and gender, and the classification of individuals as diabetic, non-diabetic, or pre-diabetic.

- **Alternate Hypothesis (H1):** There is a statistically significant correlation between factors such as age, urea, HbA1c, cholesterol, VLDL, BMI, and gender, and the classification of individuals as diabetic, non-diabetic, or pre-diabetic.

1.3 PURPOSE:

This project uses machine learning models to accurately classify patients as diabetic, non-diabetic, or pre-diabetic. This classification allows early detection and timely intervention, significantly enhancing patient outcomes and overall quality of life. By analyzing key risk factors, such as HbA1c levels, along with other health parameters, the project aims to provide healthcare professionals with effective tools to identify individuals at risk of developing diabetes. This allows personalization of treatment plans and informed decision-making in patient care. Additionally, this project aims to make a significant contribution to public health. By addressing the diabetes epidemic and assisting in the reduction of diabetic complications, it seeks to improve the overall health of populations. The utilization of machine learning in this setting represents an innovative step towards more effective, data-driven approaches in healthcare, which will lead to better health outcomes and more efficient healthcare systems.

2 METHODOLOGY

2.1 steps of the project

The purpose of this project is to create machine learning models that accurately classify patients as diabetic, non-diabetic, or pre-diabetic, enabling early detection and intervention, enhancing patient

outcomes and quality of life. Assist healthcare professionals in identifying at-risk individuals, personalizing treatment plans, and making informed decisions about patient care.
Contribute to public health by addressing the diabetes epidemic, reducing diabetic complications, and increasing overall health. Our project uses a retrospective observational method with data from (Rashid, 2020) Iraqi society that includes variables such as age, gender, BMI, blood glucose, cholesterol, and HbA1c. We aim to improve diabetes diagnosis and treatment, addressing its impact on individuals and society.

The methodology of our project involves various stages, which will be explained in more depth later in the report.

- Data Collection
- Data Preprocessing:
- Exploratory Data Analysis (EDA)
- Feature Selection
- Model Development
- Model Evaluation
- Interpretation and Reporting

2.2 original team members and responsibilities

Our team has a diverse set of skills and backgrounds. Below are the original team member responsibilities for the Project.

NAME	BACKGROUND	RESPONSIBILITIES
Pavithra Kadri	Bachelor of Medicine, Bachelor of Surgery (MBBS)	Project coordinator, data cleaning, data analysis and Research.
Deepthi mallepally	Bachelor of Dental Surgery (BDS)	Managed statistical testing and interpretation.
Deva Harsha Gubbala	BSC in microbiology and applied nutrition	Focused on exploratory data analysis and visualization.
Amulya Ragula	BSC (Hons) Agriculture	Responsible for data preprocessing and feature engineering.
Sharmila Shree V	Bachelor of Dental Surgery (BDS)	In charge of model development and evaluation.

Fig.1.original Responsibilities of Team Members for the project

2.3 Actual contribution from individual team members

Below is the Actual contribution from individual team members for the project

NAME	BACKGROUND	RESPONSIBILITIES
Pavithra Kadri	Bachelor of Medicine, Bachelor of Surgery (MBBS)	Managed data collection, ensured consistency, and executed data cleaning, addressed missing values, Project presentation and Research.
Deepthi mallepally	Bachelor of Dental Surgery (BDS)	Transformed data for analysis, identified and handled outliers, prepared data for visualization and testing.
Deva Harsha Gubbala	BSC in microbiology and applied nutrition	Developed data visualizations, explored data relationships and trends.
Amulya Ragula	BSC (Hons) Agriculture	Conducted statistical testing, assessed hypotheses, and evaluated data normality and skewness.
Sharmila Shree V	Bachelor of Dental Surgery (BDS)	Built and evaluated 12 machine learning models, analyzed performance using confusion matrix and ROC curves, compared and selected top-performing models.

Fig.2. Actual contribution from individual team members

2.4 PROJECT CHALLENGES

In our machine learning project, we encountered several challenges that required solutions and careful consideration. One significant hurdle was the imbalance in the dataset, particularly with a higher prevalence of individuals classified as diabetic (Class Y). To address this issue and ensure

a more representative training set, we implemented Synthetic Minority Over-Sampling Technique (SMOTE) to generate synthetic instances of the minority class.
Additionally, we faced challenges related to data preprocessing, as the dataset contained unique values in the 'CLASS' variable. To facilitate analysis, we had to make modifications in the Excel sheet to standardize and organize the class labels appropriately, ensuring consistency in our machine learning pipeline.
Furthermore, our dataset posed unique difficulties in terms of exploration and analysis. It was the least explored dataset, with limited references on platforms like Mendeley. The absence of established codes or prior analyses made it challenging to draw upon existing frameworks. Consequently, our team had to adopt a m methodical approach to understand the dataset's intricacies, relying on solutions for feature engineering and model development.
Despite these challenges, the project provided valuable ideas into handling imbalanced datasets, addressing unique data formatting issues, and navigating less-explored territories in machine learning research. The experience underscored the importance of adaptability and creativity when working with diverse datasets.

3.Data Collection

The dataset was obtained from the Medical City Hospital and the Specialized Center for Endocrinology and Diabetes at Al-Kindy Teaching Hospital in Iraq.

Rashid, A. (2020). Diabetes Dataset. Data.mendeley.com, 1. https://doi.org/10.17632/wj9rwkp9c2.1

3.1 Variable Descriptions:

FEATURE NAME	DESCRIPTION	DATA TYPE
Age	Age of the patient	Continuous
Urea	Urea levels in blood	Continuous
Creatinine (Cr)	Serum creatinine concentration	Continuous
HbA1c	Hemoglobin A1c levels	Continuous
Cholesterol (Chol)	Total cholesterol levels	Continuous
Triglycerides (TG)	Triglycerides concentration	Continuous
High-Density Lipoprotein (HDL)	Levels of HDL ("good" cholesterol)	Continuous
Low-Density Lipoprotein (LDL)	Levels of LDL ("bad" cholesterol)	Continuous
Very Low-Density Lipoprotein (VLDL)	Levels of VLDL	Continuous
Body Mass Index (BMI)	Body weight relative to height	Continuous
Gender	Patient's gender	Categorical
Response Variable (Target)		
Prediabetic	Elevated blood sugar, precursor to diabetes	Categorical
Diabetic	Clinically diagnosed diabetes, requiring ongoing care	Categorical
Non-Diabetic	No significant blood sugar features	Categorical

4 DATA EXTRACTION AND STORAGE

4.1 Data Extraction

We utilized Pandas in Python to load the data into a Data Frame for further analysis. We imported essential libraries for data analysis, machine learning, and deep learning, including matplotlib, seaborn, sklearn, xgboost, NumPy, pandas, scipy.io, and keras. Additionally, we included machine learning classifiers such as DecisionTreeClassifier and Logistic Regression in our project.

4.2Data Cleaning

Reading the csv data file as the Data Frame

```
df = pd.read_csv('Dataset of Diabetes PART1.csv')
```

Fig3: Reading the csv data file as the Data Frame

```
df.head()
```

	ID	No_Pation	Gender	AGE	Urea	Cr	HbA1c	Chol	TG	HDL	LDL	VLDL	BMI	CLASS
0	502	17975	F	50	4.7	46	4.9	4.2	0.9	2.4	1.4	0.5	24.0	N
1	735	34221	M	26	4.5	62	4.9	3.7	1.4	1.1	2.1	0.6	23.0	N
2	420	47975	F	50	4.7	46	4.9	4.2	0.9	2.4	1.4	0.5	24.0	N
3	680	87656	F	50	4.7	46	4.9	4.2	0.9	2.4	1.4	0.5	24.0	N
4	504	34223	M	33	7.1	46	4.9	4.9	1.0	0.8	2.0	0.4	21.0	N

Fig 4: checking for the head of the dataset

One missing value in target variable class was found in the dataset and we replaced the missing value with most frequent variable (mode).

```
Missing values count per column:
ID            0
No_Pation     0
Gender        0
AGE           0
Urea          0
Cr            0
HbA1c         0
Chol          0
TG            0
HDL           0
LDL           0
VLDL          0
BMI           0
CLASS         1
dtype: int64
```

Fig 5: checking for the missing values

```
Missing values count per column after handling:
ID           0
No_Pation    0
Gender       0
AGE          0
Urea         0
Cr           0
HbA1c        0
Chol         0
TG           0
HDL          0
LDL          0
VLDL         0
BMI          0
CLASS        0
```

Fig 6: Replacing the missing values with mode:

```python
missing_values = df.isnull()

# Display the count of missing values for each column
print("Missing values count per column:")
print(missing_values.sum())

# Display the rows with missing values
print("\nRows with missing values:")
print(df[missing_values.any(axis=1)])

# Visualize missing values using a heatmap
import seaborn as sns
plt.figure(figsize=(10, 6))
sns.heatmap(missing_values, cmap='viridis', cbar=False, yticklabels=False)
plt.title('Missing Values Heatmap')
plt.show()
```

fig7: code for checking the missing values

The dataset initially consists of 1000 entries with 14 columns, totaling approximately 109.5 KB in memory. No duplicate values are found. After removing the 'ID' and 'No_Pation' columns, since they had no significance in our analysis the dataset is left with 12 columns. The remaining columns include information on gender, age, various health indicators, BMI, and a target variable 'CLASS'. The dataset now has 844 instances of 'Y', 102 of 'N', and 53 of 'P' in the 'CLASS' variable.

```
Rows with missing values:
    ID  No_Pation Gender  AGE  Urea  Cr  HbA1c  Chol   TG  HDL  LDL  VLDL  \
21  97      12744      F   42   5.0  73    4.5   6.2  1.0  1.1  4.6   0.4

     BMI CLASS
21  24.0   NaN
```

fig 8: row showing missing values

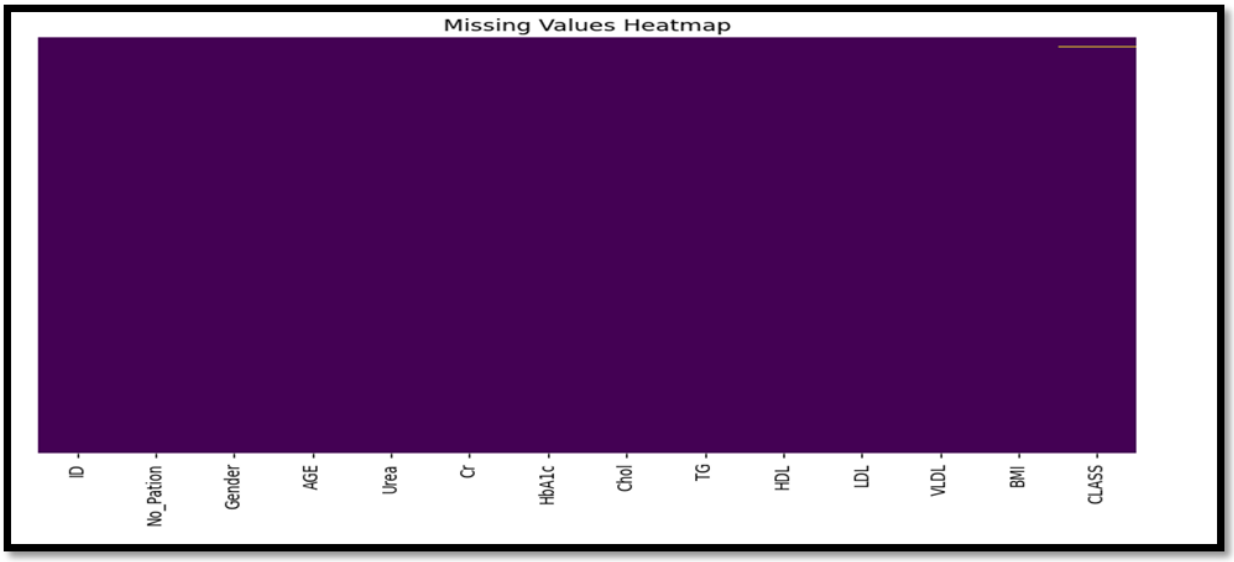

Fig 9: Heatmap Illustrating Missing Values

The heatmap visualization highlights the presence of missing values within the dataset. Specifically, a single missing value in the "CLASS" column is depicted as yellow.

```
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 1000 entries, 0 to 999
Data columns (total 14 columns):
 #   Column     Non-Null Count  Dtype
---  ------     --------------  -----
 0   ID         1000 non-null   int64
 1   No_Pation  1000 non-null   int64
 2   Gender     1000 non-null   object
 3   AGE        1000 non-null   int64
 4   Urea       1000 non-null   float64
 5   Cr         1000 non-null   int64
 6   HbA1c      1000 non-null   float64
 7   Chol       1000 non-null   float64
 8   TG         1000 non-null   float64
 9   HDL        1000 non-null   float64
 10  LDL        1000 non-null   float64
 11  VLDL       1000 non-null   float64
 12  BMI        1000 non-null   float64
 13  CLASS      999 non-null    object
dtypes: float64(8), int64(4), object(2)
memory usage: 109.5+ KB
```

Fig 10: information of the dataset:

This table provides a concise overview of the dataset, containing 1000 entries and 14 columns. The dataset includes information on various health indicators such as Patient ID (ID), Patient Number (No_Pation), Gender, Age (AGE), Urea levels (Urea), Creatinine levels (Cr), HbA1c values (HbA1c), Cholesterol levels (Chol), Triglycerides (TG), HDL and LDL cholesterol, VLDL cholesterol (VLDL), and BMI. Notably, there is a non-null count for each column, ensuring completeness of the dataset. The "CLASS" column, which denotes the classification label, contains 999 non-null entries. The data types range from integers to floating-point numbers and object types, contributing to a comprehensive dataset suitable for further analysis and machine learning model development. The memory usage is approximately 109.5 KB.

```
print("Shape of the dataset", df.shape)
```

```
Shape of the dataset (1000, 14)
```

The dataset contains 1000 entries with 14 columns The memory usage of the DataFrame is approximately 109.5 KB.

```
duplicate_rows = df[df.duplicated()]
print(duplicate_rows)
```

```
Empty DataFrame
Columns: [ID, No_Pation, Gender, AGE, Urea, Cr, HbA1c, Chol, TG, HDL, LDL, VLDL, BMI, CLASS]
Index: []
```

no duplicate values

```
df = df.drop(['ID', 'No_Pation'], axis=1)
```

removed column ID and patient number column

fig 11 checking for the shape, duplicate values, class counts and removal of columns:

```
if 'CLASS' in df.columns:
    class_counts = df['CLASS'].value_counts()


    if not class_counts.empty:
        plt.figure(figsize=(8, 6))
        class_counts.plot(kind='bar', color='pink')
        plt.title('Class Counts')
        plt.xlabel('Class')
        plt.ylabel('Count')
        plt.show()
    else:
        print("Class counts DataFrame is empty.")
else:
    print("Column 'CLASS' not found in the DataFrame.")
```

fig 12: checking for class counts code

fig 13: bar graph showing class counts

The class counts have Y, N, P with majority in Y and minority in N and P

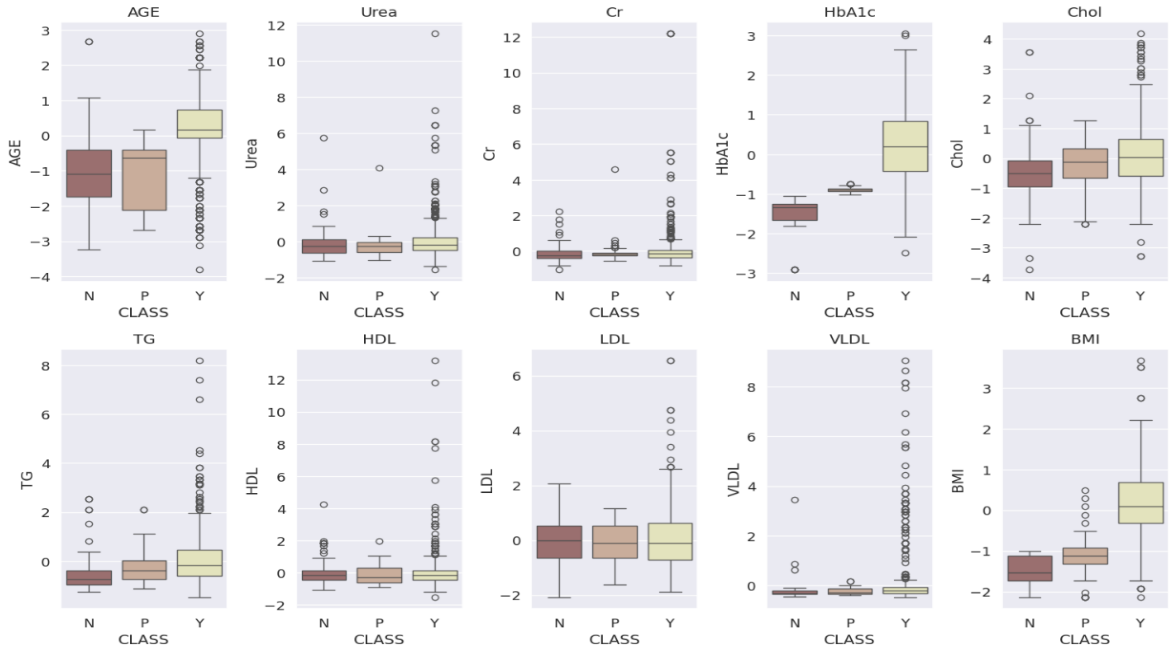

Fig 14: Before outlier removal:

The bar plots show outliers in all the variables

:

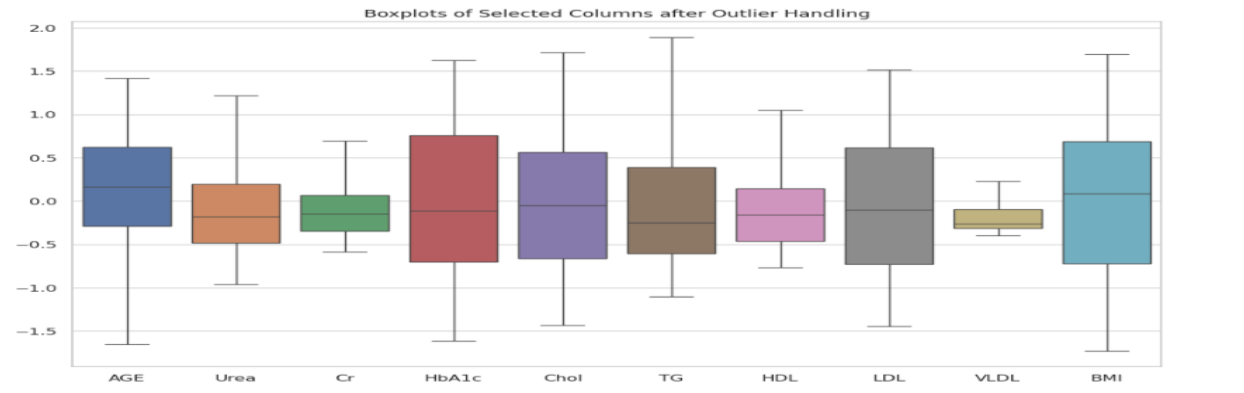

Fig 15: After replacing outlier with IQR

Outliers in the dataset were identified and replaced through an Interquartile Range (IQR), the identified outliers were replaced as part of the data preprocessing steps, ensuring a more and reliable dataset for further analysis.

Statistics	AGE	Urea	Cr	HbA1c	Chol	TG	HDL	LDL	VLDL	BMI
mean	0.035248	-0.09599	-0.10535	-0.01575	-0.0153	-0.04456	-0.08514	-0.03904	-0.19392	-0.00942
std	0.804177	0.575058	0.33079	0.926055	0.841005	0.81524	0.512955	0.853827	0.169844	0.957814
min	-1.65188	-0.96286	-0.58282	-1.61136	-1.43174	-1.10649	-0.76468	-1.44435	-0.39727	-1.72947
25%	-0.28744	-0.48565	-0.34931	-0.70326	-0.66315	-0.60666	-0.46168	-0.72657	-0.31534	-0.72139
50%	0.167371	-0.17887	-0.14916	-0.11101	-0.04828	-0.24964	-0.15869	-0.09851	-0.26072	0.085078
75%	0.622183	0.196086	0.067668	0.757616	0.566588	0.393002	0.1443	0.619276	-0.09687	0.689928
max	1.418104	1.218686	0.693139	1.626241	1.719471	1.892493	1.053276	1.516504	0.230845	1.698012

Fig 16: Descriptive statistics of our data set:

The summary statistics reveal that the dataset contains 1000 entries for each variable. The data has been standardized, with mean values close to zero and standard deviations around one. The minimum and maximum values indicate the range of the variables, while quartiles provide ideas into the data distribution, the dataset exhibits consistent scaling and shows no apparent anomalies in the summary statistics

5 Data visualization

```
import matplotlib.pyplot as plt
import seaborn as sns
import numpy as np

numeric_columns = df.select_dtypes(include=[np.number]).columns


fig, axes = plt.subplots(nrows=len(numeric_columns), ncols=1, figsize=(10, 5 * len(numeric_columns)))

for i, col in enumerate(numeric_columns):
    sns.histplot(df[col], ax=axes[i], kde=True, bins=20, color='red')
    axes[i].set_title(f'Distribution of {col}', color='red')  # Set title color to red

plt.tight_layout()
plt.show()
```

Fig 17: Histogram code

5.1 Histogram of all the variables

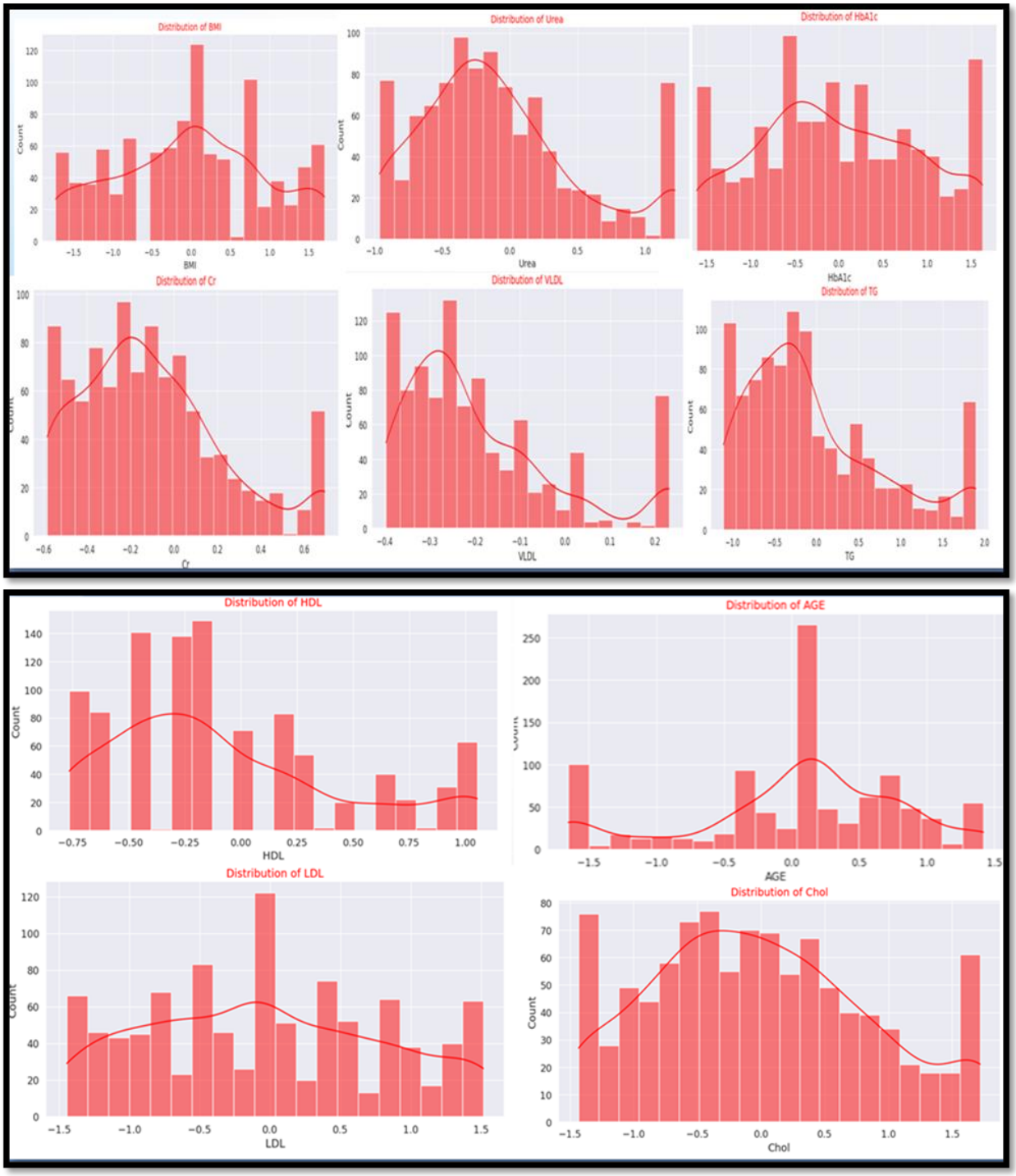

fig 18: histogram visualization

Distribution of BMI

•The negatively skewed distribution suggests that a larger portion of the population falls within the healthy or overweight range (BMI 18.5-24.9 and 25-29.9, respectively), while smaller percentage falls underweight (BMI < 18.5) or obese (BMI > 30).

Distribution of Creatinine and Urea

•Similar to VLDL and LDL, these histograms exhibit positive skews, potentially indicating early signs of kidney problems for individuals with elevated Creatinine and Urea levels. These are waste products filtered by the kidneys, and elevated levels can suggest impaired kidney function.

Distribution of HbA1c

•This histogram shows a positive-skewed distribution with a potential tail towards higher values. While the peak suggests a majority with HbA1c levels within the normal range, the tail indicates individuals who might be prediabetic or diabetic. HbA1c reflects long-term blood sugar control, and elevated levels are associated with diabetes and its complications.

Distribution of VLDL, LDL, Chol, and TG

•These histograms share similar positive-skewed patterns, indicating that while most individuals have VLDL, LDL, Cholesterol, and Triglyceride levels within a healthy range, there are also some with higher values. These elevated levels are risk factors for cardiovascular diseases like heart attack and stroke.

Distribution of HDL

•This histogram stands out with a more positively skewed. This is a positive sign, as HDL cholesterol is considered "good" cholesterol and helps remove LDL from the arteries. Higher HDL levels are associated with a lower risk of heart disease.

Distribution of Age

•The negatively skewed distribution suggests a focus on a middle-aged to older adult population. This aligns with the age groups most susceptible to chronic diseases like those potentially reflected in the other histograms.

```
import matplotlib.pyplot as plt
from scipy.stats import gaussian_kde

variables_to_plot = [ 'BMI', 'HbA1c', 'AGE']


num_columns = len(variables_to_plot)
num_rows = 1


plt.figure(figsize=(15, 5 * num_rows))

for i, variable in enumerate(variables_to_plot):
    plt.subplot(num_rows, num_columns, i + 1)

    # Calculate the kernel density estimate (KDE) for 'CLASS' categories
    kde_class_N = gaussian_kde(df[df['CLASS'] == 'N'][variable])
    kde_class_P = gaussian_kde(df[df['CLASS'] == 'P'][variable])
    kde_class_Y = gaussian_kde(df[df['CLASS'] == 'Y'][variable])

    x = np.linspace(df[variable].min(), df[variable].max(), 100)  # Adjust the
    y_class_N = kde_class_N(x)
    y_class_P = kde_class_P(x)
    y_class_Y = kde_class_Y(x)


    plt.fill_between(x, y_class_N, alpha=0.4, label='CLASS=N', color='blue')


    plt.fill_between(x, y_class_P, alpha=0.4, label='CLASS=P', color='red')


    plt.fill_between(x, y_class_Y, alpha=0.4, label='CLASS=Y', color='green')

    plt.title(f'KDE plot of {variable} by CLASS')
    plt.xlabel(variable)
    plt.ylabel('Density')
```

```
    print(f"Variable: {variable}")
    print("Mean values:")
    print(f"CLASS=N: {df[df['CLASS'] == 'N'][variable].mean()}")
    print(f"CLASS=P: {df[df['CLASS'] == 'P'][variable].mean()}")
    print(f"CLASS=Y: {df[df['CLASS'] == 'Y'][variable].mean()}")
    print("\n")

plt.tight_layout()
plt.show()
```

Fig 19: KDE PLOT code

```
variables_to_plot = [ 'Urea', 'Cr', 'Chol',]


num_columns = len(variables_to_plot)
num_rows = 1


plt.figure(figsize=(15, 5 * num_rows))

for i, variable in enumerate(variables_to_plot):
    plt.subplot(num_rows, num_columns, i + 1)


    kde_class_N = gaussian_kde(df[df['CLASS'] == 'N'][variable])
    kde_class_P = gaussian_kde(df[df['CLASS'] == 'P'][variable])
    kde_class_Y = gaussian_kde(df[df['CLASS'] == 'Y'][variable])

    x = np.linspace(df[variable].min(), df[variable].max(), 100)
    y_class_N = kde_class_N(x)
    y_class_P = kde_class_P(x)
    y_class_Y = kde_class_Y(x)


    plt.fill_between(x, y_class_N, alpha=0.4, label='CLASS=N', color='blue')


    plt.fill_between(x, y_class_P, alpha=0.4, label='CLASS=P', color='red')


    plt.fill_between(x, y_class_Y, alpha=0.4, label='CLASS=Y', color='green')

    plt.title(f'KDE plot of {variable} by CLASS')
    plt.xlabel(variable)
    plt.ylabel('Density')
    plt.legend()
```

```
    print(f"Variable: {variable}")
    print("Mean values:")
    print(f"CLASS=N: {df[df['CLASS'] == 'N'][variable].mean()}")
    print(f"CLASS=P: {df[df['CLASS'] == 'P'][variable].mean()}")
    print(f"CLASS=Y: {df[df['CLASS'] == 'Y'][variable].mean()}")
    print("\n")

plt.tight_layout()
plt.show()
```

Fig 20: KDE PLOT code

```
import plotly.express as px
px.defaults.template = "plotly_dark"
colors = {'N': 'blue', 'Y': 'green', 'P': 'red'}
fig_bmi = px.scatter(df, x='BMI', color='CLASS', title='BMI Scatter Plot', color_discrete_map=colors)
fig_bmi.update_traces(marker=dict(size=10))
fig_bmi.show()
fig_chol = px.scatter(df, x='AGE', y='Chol', color='CLASS', title='Cholesterol Scatter Plot', color_discrete_map=colors)
fig_chol.update_traces(marker=dict(size=10))
fig_chol.show()


fig_hba1c = px.scatter(df, x='HbA1c', color='CLASS', title='HbA1c Scatter Plot', color_discrete_map=colors)
fig_hba1c.update_traces(marker=dict(size=10))
fig_hba1c.show()


fig_urea = px.scatter(df, x='Urea', color='CLASS', title='Urea Scatter Plot', color_discrete_map=colors)
fig_urea.update_traces(marker=dict(size=10))
fig_urea.show()


fig_cr = px.scatter(df, x='Cr', color='CLASS', title='Creatinine Scatter Plot', color_discrete_map=colors)
fig_cr.update_traces(marker=dict(size=10))
fig_cr.show()


fig_tg = px.scatter(df, x='TG', color='CLASS', title='Triglycerides Scatter Plot', color_discrete_map=colors)
fig_tg.update_traces(marker=dict(size=10))
fig_tg.show()


fig_hdl = px.scatter(df, x='HDL', color='CLASS', title='HDL Scatter Plot', color_discrete_map=colors)
fig_hdl.update_traces(marker=dict(size=10))
fig_hdl.show()


fig_ldl = px.scatter(df, x='LDL', color='CLASS', title='LDL Scatter Plot', color_discrete_map=colors)
fig_ldl.update_traces(marker=dict(size=10))
fig_ldl.show()

fig_vldl = px.scatter(df, x='Gender', color='CLASS', title='gender Scatter Plot', color_discrete_map=colors)
fig_vldl.update_traces(marker=dict(size=10))
fig_vldl.show()



fig_vldl = px.scatter(df, x='VLDL', color='CLASS', title='VLDL Scatter Plot', color_discrete_map=colors)
fig_vldl.update_traces(marker=dict(size=10))
fig_vldl.show()
```

Fig 21: Scatter plot correlation code

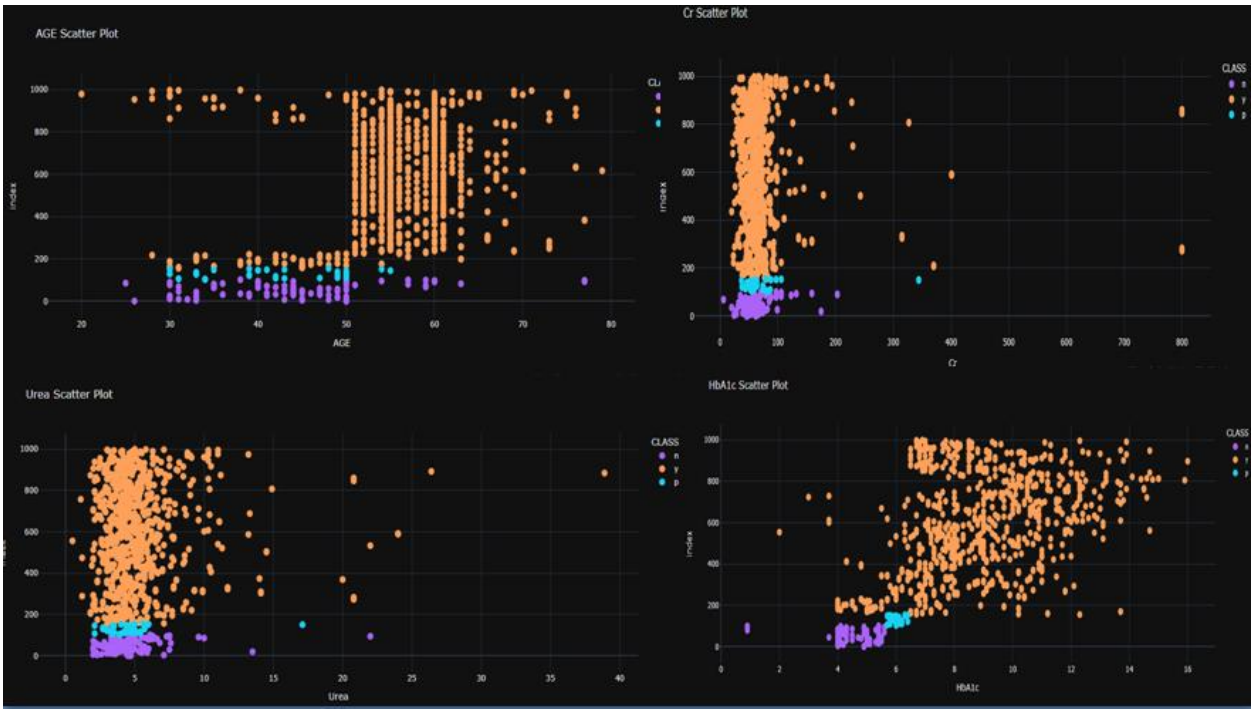

Fig 22.1: scatterplot of AGE, Cr, urea and HbA1c

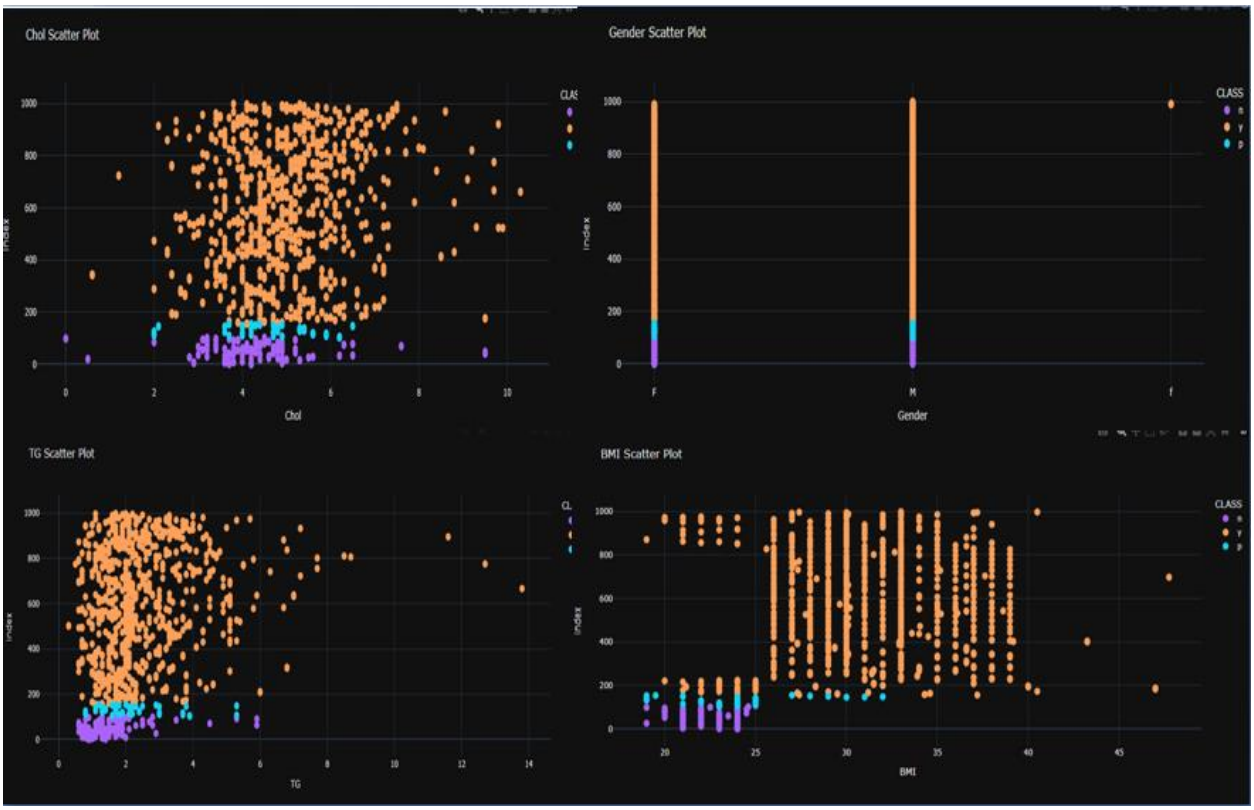

Fig 22.2: scatterplot of Chol, Gender, TG, BMI

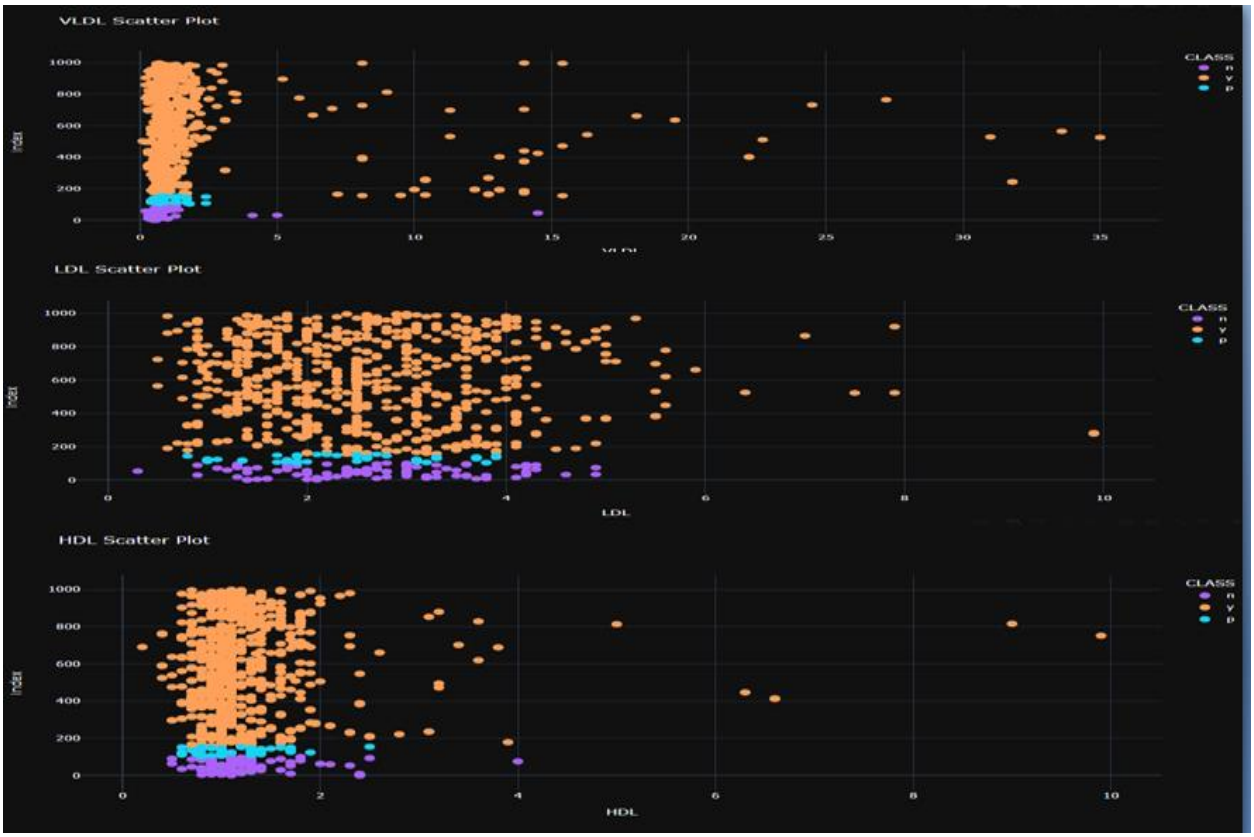

Fig22.3: Scatter plot visualization of VLDL, LDL, HDL

AGE Scatter Plot:

- Class 'y' has an unexpectedly high mean age compared to 'n' and 'p', potentially indicating outliers.

Urea Scatter Plot:

- Class 'y' shows significantly higher mean Urea values compared to 'n' and 'p', suggesting a potential distinguishing factor.

Cr Scatter Plot:

- Like Urea, Class 'y' exhibits notably higher mean Creatinine values.

HbA1c Scatter Plot:

- Class 'y' displays higher mean HbA1c values, indicating potential differences in glycated hemoglobin levels.

Chol Scatter Plot:

- Class 'y' shows higher mean Cholesterol values, potentially associated with the disease class.

TG Scatter Plot:

- Class 'y' exhibits higher mean Triglyceride levels compared to 'n' and 'p.'

BMI Scatter Plot:

- Class 'y' has a notably higher mean BMI, potentially correlating with the disease class.

Lipid-Related Scatter Plots (HDL, LDL, VLDL):

- Class 'y' consistently shows higher mean values compared to other classes.

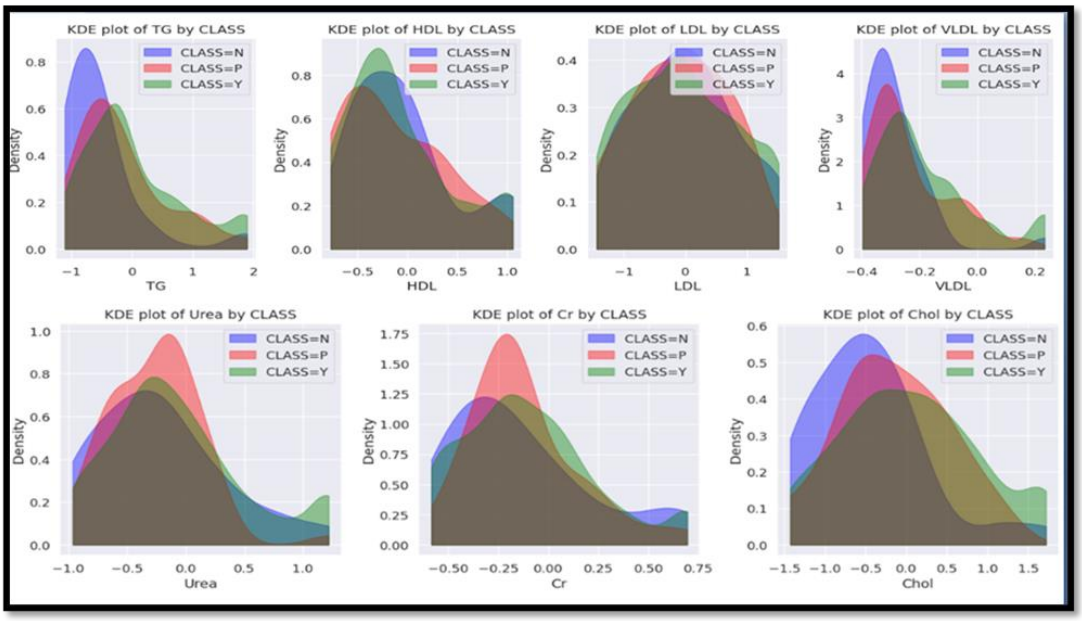

Fig 23.1: Bivariate Analysis correlation matrix code

Urea

- Individuals with CLASS=Y tend to have higher Urea levels compared to CLASS=N and CLASS=P.

Cr

- Individuals with CLASS=Y tend to have higher Creatinine levels compared to CLASS=P and CLASS=N.

Chol

- Individuals with CLASS=Y tend to have higher cholesterol levels compared to CLASS=P and CLASS=N.

TG

- Individuals with CLASS=Y tend to have higher Triglyceride levels compared to CLASS=P and CLASS=N.

HDL

- Individuals with CLASS=Y may have slightly higher HDL levels compared to CLASS=N and CLASS=P.

LDL

- Individuals with CLASS=Y may have slightly higher LDL levels compared to CLASS=N and CLASS=P.

VLDL

- Individuals with CLASS=Y tend to have higher VLDL levels compared to CLASS=P and CLASS=N.

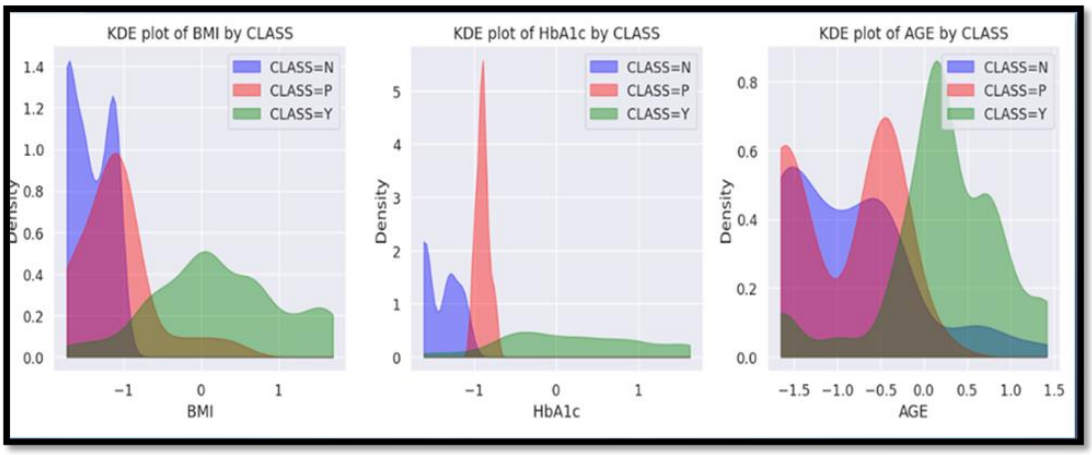

Fig 23.2: Bivariate Analysis correlation matrix code

BMI

- Individuals with CLASS=Y tend to have higher BMI compared to CLASS=P and CLASS=N.

HbA1c

- Individuals with CLASS=Y have higher HbA1c levels compared to CLASS=P and CLASS=N.

AGE

- Individuals with CLASS=Y are generally older compared to CLASS=P and CLASS=N.

These are interpretations for bivariate analysis

```
columns = ["CLASS", "AGE", "Urea", "Cr", "HbA1c", "Chol", "TG", "HDL", "LDL", "VLDL", "BMI", "Gender"]
data = df[columns]


data['CLASS'] = data['CLASS'].str.strip()


color_palette = sns.color_palette("husl", n_colors=len(data['CLASS'].unique()))


sns.set(font_scale=1.2)
plt.figure(figsize=(16, 12))

# Create a pairplot
pairplot = sns.pairplot(data, hue='CLASS', palette=color_palette, diag_kind='kde')

# Interpretation
plt.suptitle('Pairplot with Class Colors', y=1.02)
plt.show()

for i, var1 in enumerate(columns[1:]):
    for j, var2 in enumerate(columns[1:]):
        if i != j:
            plt.figure(figsize=(8, 6))
            sns.scatterplot(data=data, x=var1, y=var2, hue='CLASS', palette=color_palette)
            plt.title(f'Scatter Plot: {var1} vs {var2} with Class Colors')
            plt.show()
```

```
import seaborn as sns
import matplotlib.pyplot as plt
from sklearn.preprocessing import OneHotEncoder

# Encode only the 'Gender' column
df_encoded = pd.get_dummies(df, columns=['Gender'])

# Exclude non-numeric columns and 'CLASS'
numeric_df = df_encoded.select_dtypes(include=[np.number])

# Compute correlation matrix
corr_matrix = numeric_df.corr()

# Plot correlation heatmap
plt.figure(figsize=(12, 10))
cmap = sns.color_palette("rainbow", as_cmap=True)
sns.heatmap(corr_matrix, annot=True, cmap=cmap, vmin=-1, vmax=1, center=0, linewidths=.5)
plt.title('Correlation Heatmap')
plt.show()
```

fig 24: code for correlation heatmap

Fig25: Pair plots and Heat map correlation

•Positive Correlations:

BMI

- As BMI increases, AGE, HbA1c, TG, LDL, and VLDL tend to increase as well.

HbA1c and AGE

- There is a moderate positive relationship, indicating that as AGE increases, HbA1c tends to increase.

Cholesterol (Chol)

- Shows positive correlations with LDL and TG, suggesting that as Cholesterol increases, LDL and TG also tend to increase.

Negative Correlations

HDL

- Exhibits a negative correlation with LDL and TG. As HDL increases, LDL and TG tend to decrease.

Moderate Correlations

Urea and Cr

- Shows a moderate positive correlation, implying a relatively strong positive relationship between these two variables.

Interpretations of heatmap:

- ·The strong positive correlation between age and urea and creatinine suggests that these markers of kidney function tend to decline with age.
- ·The strong positive correlation between HbA1c and cholesterol suggests that people with higher blood sugar levels are also more likely to have high cholesterol levels.
- ·The strong positive correlation between TG and cholesterol, and between LDL and cholesterol, suggests that these lipids are often elevated together.
- ·The strong positive correlation between BMI and LDL and VLDL suggests that people with higher BMI levels are more likely to have elevated levels of these harmful lipids.
- ·The strong negative correlation between HDL, TG, and VLDL suggests that HDL cholesterol has a protective effect against the accumulation of these harmful lipids.

6 .Statistical testing

Test	Variable	Statistic	p-value	Normality
Shapiro-Wilk	Age	0.9303	3.05E-21	No
	Urea	0.9373	3.51E-20	No
	Cr	0.9445	5.92E-19	No
	HbA1c	0.9672	2.40E-14	No
	Chol	0.9713	3.68E-13	No
	TG	0.9051	1.76E-24	No
	HDL	0.9131	1.60E-23	No
	LDL	0.9632	2.89E-15	No
	VLDL	0.87	4.76E-28	No
	BMI	0.9684	4.98E-14	No
Anderson-Darling	Age	23.09	-	No
	Urea	15.72	-	No
	Cr	11.69	-	No
	HbA1c	6.09	-	No
	Chol	4.87	-	No
	TG	28.66	-	No
	HDL	26.44	-	No

	LDL	**7.48**	**-**	**No**
	VLDL	**39.27**	**-**	**No**
	BMI	**6.14**	**-**	**No**

Fig 26: Shapiro wilk and Anderson test values:

6.1Shapiro test

```
from scipy.stats import shapiro

variables_of_interest = ['AGE', 'Urea', 'Cr', 'HbA1c', 'Chol', 'TG', 'HDL', 'LDL', 'VLDL', 'BMI']

for variable in variables_of_interest:
    stat, p_value = shapiro(df[variable])
    print(f"Shapiro-Wilk test for {variable}: Statistic = {stat}, p-value = {p_value}")

    if p_value > 0.05:
        print(f"The data for {variable} appears to be normally distributed.")
    else:
        print(f"The data for {variable} does not appear to be normally distributed.")
    print("\n")
```

fig 27: Shapiro test code

6.2 Anderson darling test

```
from scipy.stats import anderson

variables_of_interest = ['AGE', 'Urea', 'Cr', 'HbA1c', 'Chol', 'TG', 'HDL', 'LDL', 'VLDL', 'BMI']

for variable in variables_of_interest:
    result = anderson(df[variable])
    print(f"Anderson-Darling test for {variable}: Statistic = {result.statistic}, Critical Values = {result.critical_values}

    if result.statistic < result.critical_values[2]:
        print(f"The data for {variable} appears to be normally distributed.")
    else:
        print(f"The data for {variable} does not appear to be normally distributed.")
    print("\n")
```

fig 28: Anderson test code

The p-value, falling below the 0.05 threshold, serves as a crucial indicator pointing towards non-normality in our dataset. This result implies that the distribution of all variables under consideration deviates significantly from a normal distribution. This finding holds relevance in guiding subsequent analyses, justifying the application of non-parametric testing methods, and ensuring a more accurate understanding of our dataset's characteristics.

In the Shapiro and Anderson test, all the variables are seen not normally distributed so we proceeded with non-parametric testing to know the significance of the variables.

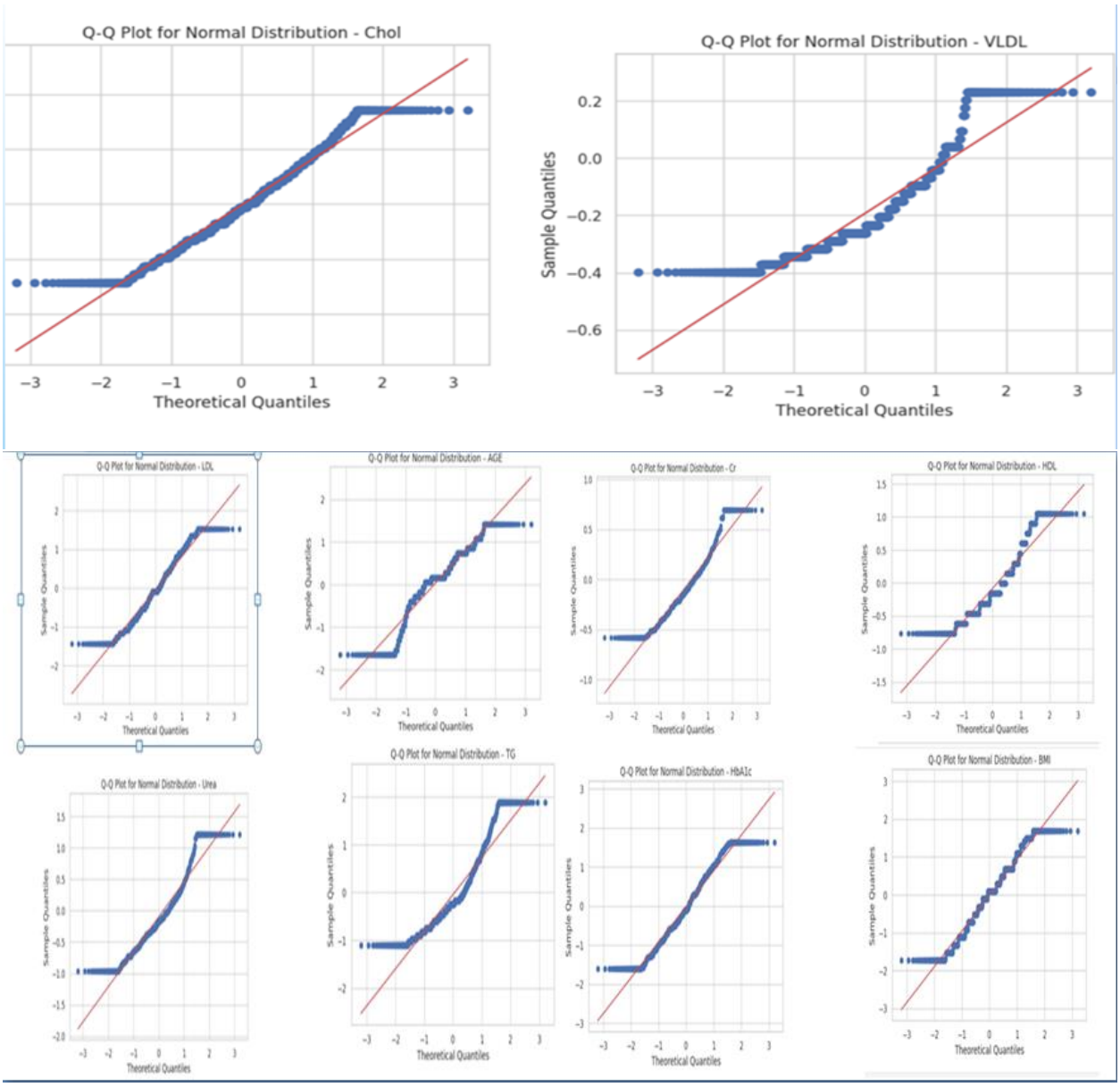

fig 29: QQ plots

These plots provide strong visual validation of non-normality and reveals large mismatches between sample vs theoretical it Confirms conclusions from statistical normality tests performed

6.3 Skewness test

:

Variable	Skewness
Age	-0.6133
Urea	0.733
Cr	0.7009
HbA1c	0.0965
Chol	0.298
TG	0.9308
HDL	0.7765
LDL	0.1539
VLDL	1.1632
BMI	-0.0181

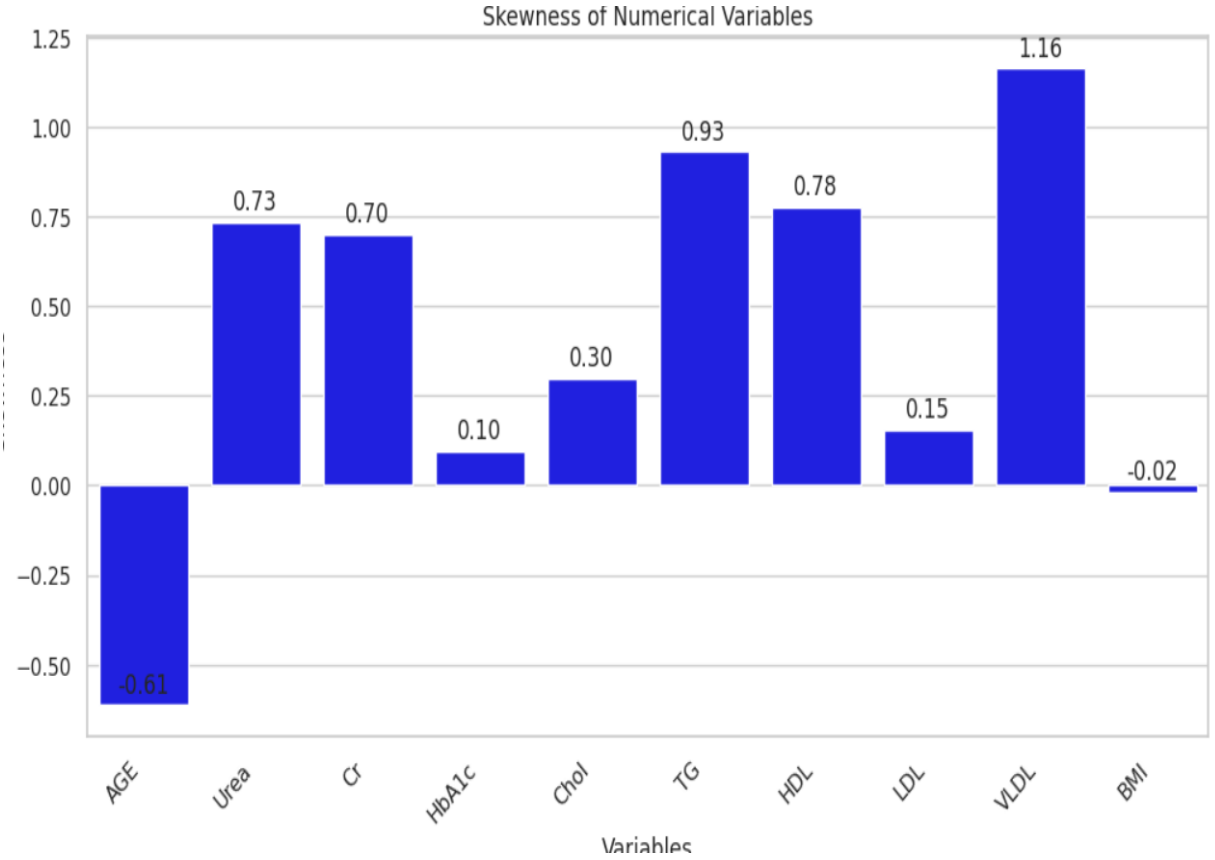

fig 30: skewness test visualization

```
# Selecting only numerical columns for skewness calculation
numerical_columns = df.select_dtypes(include=['float64', 'int64']).columns

# Calculate skewness for each numerical column
skewness_values = df[numerical_columns].skew()

# Print the skewness values
print("Skewness Values:")
print(skewness_values)

# Plotting the skewness values with annotations
plt.figure(figsize=(12, 6))
ax = sns.barplot(x=skewness_values.index, y=skewness_values.values, color='blue')
plt.title('Skewness of Numerical Variables')
plt.xlabel('Variables')
plt.ylabel('Skewness')
plt.xticks(rotation=45, ha='right')

# Adding numeric values as annotations on the bars
for i, v in enumerate(skewness_values.values):
    ax.text(i, v + 0.02, f'{v:.2f}', ha='center', va='bottom')

plt.show()
```

fig 31: skewness test code

The skewness test outcomes reveal a noteworthy insight: Values close to 0 would typically indicate a normal distribution, but our results indicate some variables exhibit skewness. This departure from a skewness value near zero strongly implies a non-normal distribution for those specific variables. The presence of skewness suggests that the data in these variables may be asymmetrical and deviate from the symmetrical bell-shaped curve characteristic of a normal distribution. This information is crucial for acknowledging the distributional characteristics of the data and further underscores the appropriateness of opting for non-parametric testing methods considering non-normality.

6.4 Chi-square test:

Variable	p-value
HbA1c	8.25E-73
BMI	6.55E-71
Age	1.67E-50
TG	6.46E-14

VLDL	2.94E-12
Chol	6.89E-09
Gender	0.000166
Urea	0.0138
Cr	0.29
HDL	0.556
LDL	0.886

fig 32: chi-square test values

```
contingency_table = pd.crosstab(df['Gender'], df['CLASS'])


chi2, p, dof, expected = chi2_contingency(contingency_table)


print(f"Chi-square value: {chi2}")
print(f"P-value: {p}")
print(f"Degrees of freedom: {dof}")
print("Expected frequencies table:")
print(expected)
```

```
Chi-square value: 17.420839096251147
P-value: 0.00016485907364661626
Degrees of freedom: 2
Expected frequencies table:
[[ 44.37   23.055 367.575]
 [ 57.63   29.945 477.425]]
```

fig 33: chi square test code

```python
# Assuming 'df' is your DataFrame
observed = pd.crosstab(df['Gender'], df['CLASS'])

# Chi-square test
chi2, p, _, _ = chi2_contingency(observed)

# Number of observations
n = np.sum(observed)

# Calculate Cramér's V
cramers_v = np.sqrt(chi2 / (n * (min(observed.shape) - 1)))

print(f"Chi-square value: {chi2}")
print(f"P-value: {p}")
print(f"Cramér's V: {cramers_v.iloc[0]:.4f}")
```

```
Chi-square value: 17.420839096251147
P-value: 0.00016485907364661626
Cramér's V: 0.4133
```

The obtained p-value, which is less than 0.05, signifies a statistically significant association between gender and diabetes status. This result reinforces the observed association, indicating that the relationship between gender and diabetes status is not likely due to random chance and holds statistical significance.

fig 34: Cramer's code

Class Y	Class N	Class P	Observed Frequencies	Expected Frequencies
Yes	No	Total	44.37	57.63
23.055	29.945			
367.575	477.425			

Fig 35: Table for values for Cramer v test

After conducting Cramér's V test, we obtained a value of 0.4133, indicating a moderate association between gender and diabetes status. This result underscores that gender does exhibit a discernible link with diabetes status, providing quantitative support to the observed association.

```
from scipy.stats import kruskal


class_n = df[df['CLASS'] == 'N']
class_y = df[df['CLASS'] == 'Y']
class_p = df[df['CLASS'] == 'P']


variables_to_test = ['AGE', 'Urea', 'Cr', 'HbA1c', 'Chol', 'TG', 'HDL', 'LDL', 'VLDL', 'BMI','Gender']


p_values = {}


for variable in variables_to_test:
    h_statistic, p_value = kruskal(class_n[variable], class_y[variable], class_p[variable])
    p_values[variable] = p_value


sorted_variables = sorted(p_values, key=p_values.get)


print("Variables in order of significance (from highly significant to lowest):")
for i, variable in enumerate(sorted_variables):
    print(f"{i+1}. {variable}: p-value = {p_values[variable]}")


plt.figure(figsize=(10, 6))
bar_colors = ['pink' if p_values[variable] < 0.05 else 'lightblue' for variable in sorted_variables]
bars = plt.bar(range(len(p_values)), [p_values[variable] for variable in sorted_variables], align='center', color=bar_colors)
plt.xticks(range(len(p_values)), sorted_variables, rotation=45)
plt.xlabel('Variables')
plt.ylabel('P-value')
plt.title('Variable Significance (Kruskal-Wallis Test)')
```

```
plt.axhline(y=0.05, color='red', linestyle='--', label='0.05 Significance Level')
plt.axhline(y=0.01, color='green', linestyle='--', label='0.01 Significance Level')


for bar, color in zip(bars, bar_colors):
    height = bar.get_height()
    plt.text(bar.get_x() + bar.get_width() / 2, height + 0.01, f"{height:.3f}", ha='center', va='bottom', color=color)


plt.legend()

plt.show()
```

fig 36: Kruskal Wallis code

6.5Variables in order of significance (from highly significant to lowest):

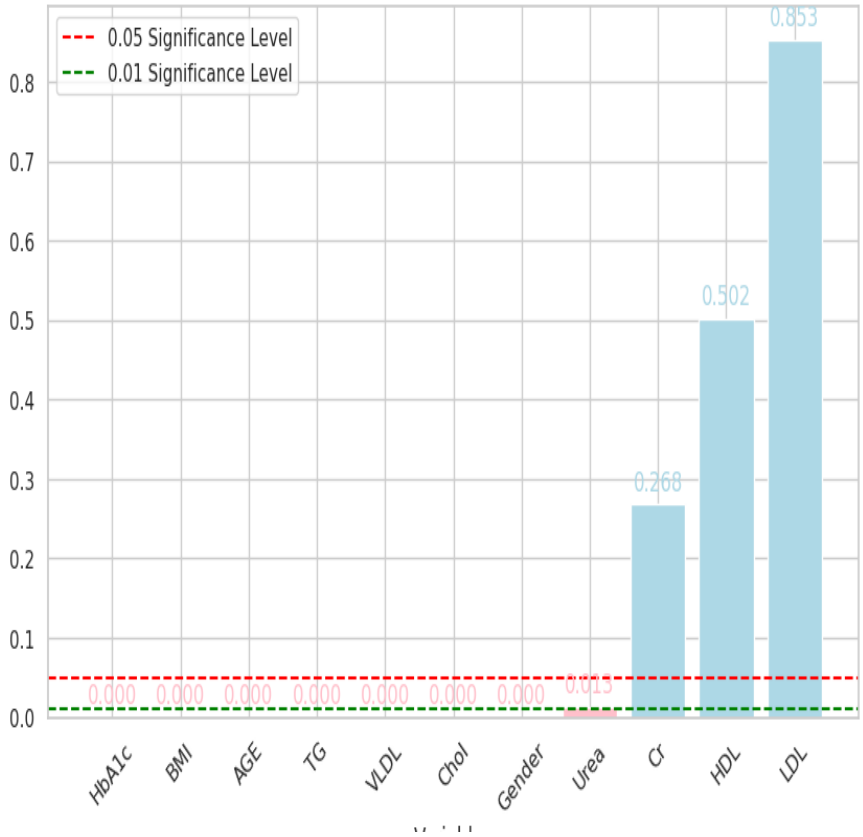

Fig 37: Variables in order of significance (from highly significant to lowest):

Rank	Variable	p-value
1	HbA1c	8.25E-73
2	BMI	6.55E-71
3	AGE	1.67E-50
4	TG	6.46E-14
5	VLDL	2.94E-12
6	Chol	6.89E-09
7	Gender	0.000166
8	Urea	0.01389
9	Cr	0.2901
10	HDL	0.5561
11	LDL	0.886

Fig 38: Table for Variables in order of significance (from highly significant to lowest)

Our findings across various variables reveal substantial differences among the classes. Notably, HbA1c and BMI exhibited the most significant distinctions, underscoring their role in delineating the three groups and emphasizing their high clinical importance. Age, TG, VLDL, and Cholesterol also displayed highly significant p-values, suggesting considerable variations across the classes. Urea and Gender, while presenting more modest but still significant p-values, indicate discernible differences in Urea levels across groups, with gender contributing to these variations. Conversely, Creatinine (Cr), High-Density Lipoprotein (HDL), and Low-Density Lipoprotein (LDL) showed non-significant differences between classes, leading to their removal to streamline the dataset.

7. Feature Engineering and Variable Selection

```
columns_to_drop = ['Cr', 'HDL', 'LDL']
df_filtered = df.drop(columns=columns_to_drop)

# Display the filtered DataFrame
print(df_filtered)

features = ["Gender", "AGE", "Urea", "HbA1c", "Chol", "TG", "VLDL", "BMI"]
label_encoder = LabelEncoder()

# Encode categorical variables (assuming 'Gender' is categorical)
df['Gender'] = label_encoder.fit_transform(df['Gender'])
df['CLASS'] = label_encoder.fit_transform(df['CLASS'])

X = df[features]
y = df["CLASS"]
```

fig 39: code for dropping columns Cr, HDL, and LDL

In optimizing the predictive model, HDL, LDL, and CR variables were excluded due to low significance. Feature engineering was then applied to enhance the remaining variables, improving overall model efficiency and interpretability.

7.1 SMOTE (Synthetic Minority Over-sampling Technique)

SMOTE was utilized to generate synthetic samples for the minority class ('P'). The implementation involved creating synthetic instances of the minority class by interpolating between existing samples. The sampling strategy parameter was set to 'auto' to automatically balance the class distribution, and the k_neighbors parameter was set to 3 to determine the number of nearest neighbors used in the

interpolation process.

```
from imblearn.over_sampling import SMOTE
```

```
smote = SMOTE(sampling_strategy='auto', k_neighbors=3, random_state=42)
```

```
from imblearn.over_sampling import RandomOverSampler

ros = RandomOverSampler(sampling_strategy='auto', random_state=42)
X_resampled, y_resampled = ros.fit_resample(X_train, y_train)
```

fig 40: code for SMOTE

This analysis employed essential Python libraries: NumPy for numerical operations, Matplotlib for visualization, and scikit-learn for machine learning tools (e.g., train_test_split, confusion matrix). Various scikit-learn classifiers were used, and class imbalance was tackled with imbalanced-Learn's SMOTE. This streamlined toolkit enabled efficient data handling, model training, and evaluation.

```
import pandas as pd
from sklearn.model_selection import train_test_split
from sklearn.preprocessing import StandardScaler
from sklearn.metrics import accuracy_score, classification_report
from sklearn.model_selection import cross_val_score
from sklearn.ensemble import RandomForestClassifier
from sklearn.svm import SVC
from sklearn.neighbors import KNeighborsClassifier
from sklearn.linear_model import LogisticRegression
from sklearn.tree import DecisionTreeClassifier
from xgboost import XGBClassifier
from sklearn.ensemble import GradientBoostingClassifier
from sklearn.naive_bayes import GaussianNB
from sklearn.neural_network import MLPClassifier
from sklearn.ensemble import AdaBoostClassifier
from sklearn.ensemble import BaggingClassifier
from sklearn.ensemble import ExtraTreesClassifier
from sklearn.pipeline import make_pipeline
from sklearn.impute import SimpleImputer
```

fig 41: code for importing

A diverse set of 12 classifiers was employed to evaluate their performance on the given dataset. The classifiers included Logistic Regression, Support Vector Machine (SVM), K-Nearest Neighbors (KNN),

Random Forest, Decision Tree, XGBoost, Gradient Boosting, Gaussian Naive Bayes, Multi-layer Perceptron (MLP), AdaBoost, Bagging, and Extra Trees classifiers.

Each classifier was trained on the training dataset, and predictions were made on the test dataset. Performance metrics such as accuracy and classification report were computed.

```
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state=42)

classifiers = [
    LogisticRegression(max_iter=1000, random_state=42),
    SVC(random_state=42),
    KNeighborsClassifier(),
    RandomForestClassifier(random_state=42),
    DecisionTreeClassifier(random_state=42),
    XGBClassifier(random_state=42),
    GradientBoostingClassifier(random_state=42),
    GaussianNB(),
    MLPClassifier(max_iter=1000, random_state=42),
    AdaBoostClassifier(random_state=42),
    BaggingClassifier(random_state=42),
    ExtraTreesClassifier(random_state=42)
]

pipeline = make_pipeline(SimpleImputer(strategy='mean'), StandardScaler())

for classifier in classifiers:
    clf = classifier

    if isinstance(classifier, (MLPClassifier)):
        clf = make_pipeline(pipeline, classifier)

    clf.fit(X_train, y_train)

    y_pred = clf.predict(X_test)
```

```
accuracy = accuracy_score(y_test, y_pred)
classification_rep = classification_report(y_test, y_pred, zero_division=1)

print(f"\nClassifier: {type(classifier).__name__}")
print(f"Accuracy: {accuracy:.2%}")
print("Classification Report:")
print(classification_rep)


try:
    cross_val_scores = cross_val_score(clf, X, y, cv=5, scoring='accuracy')
    print(f"Cross-Validation Accuracy: {cross_val_scores.mean():.2%}")
except Exception as e:
    print(f"Cross-Validation Error: {str(e)}")
```

fig 42: code for model performance

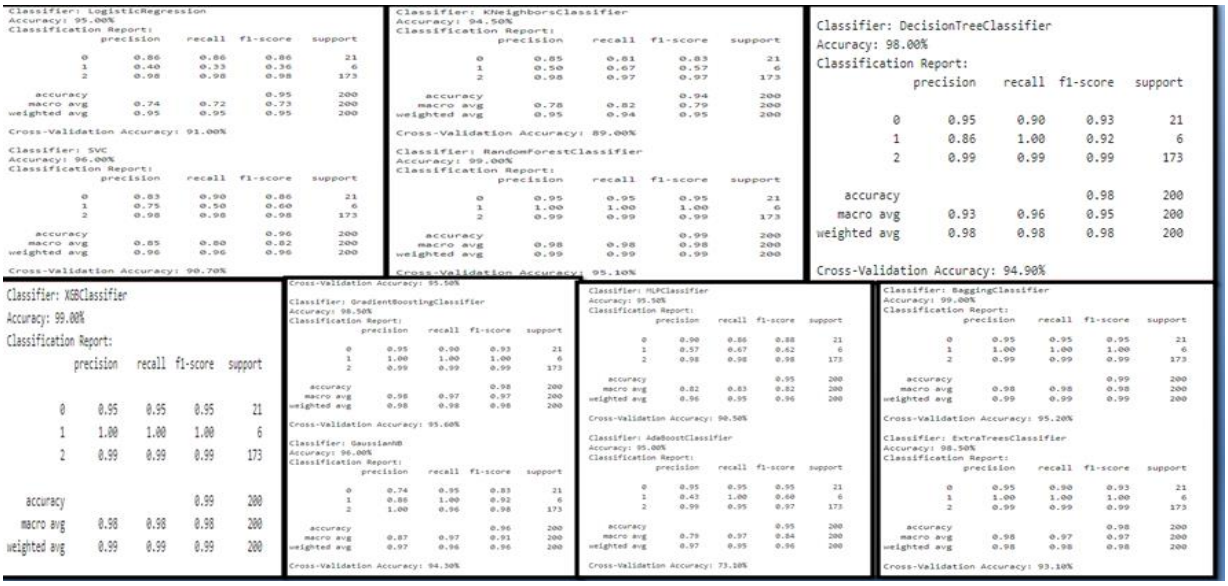

fig 43: output for all model performance

In our extensive assessment of machine learning classifiers for multi-class classification, models like Random Forest, XGBoost, and Bagging stood out with an impressive 99.00% accuracy. These top-performing models consistently demonstrated high precision, recall, and F1-scores across cross-validation folds. Conversely, the AdaBoost Classifier, while achieving lower overall accuracy, excelled in precision and recall for specific classes.

7.2 Confusion matrix for all models:

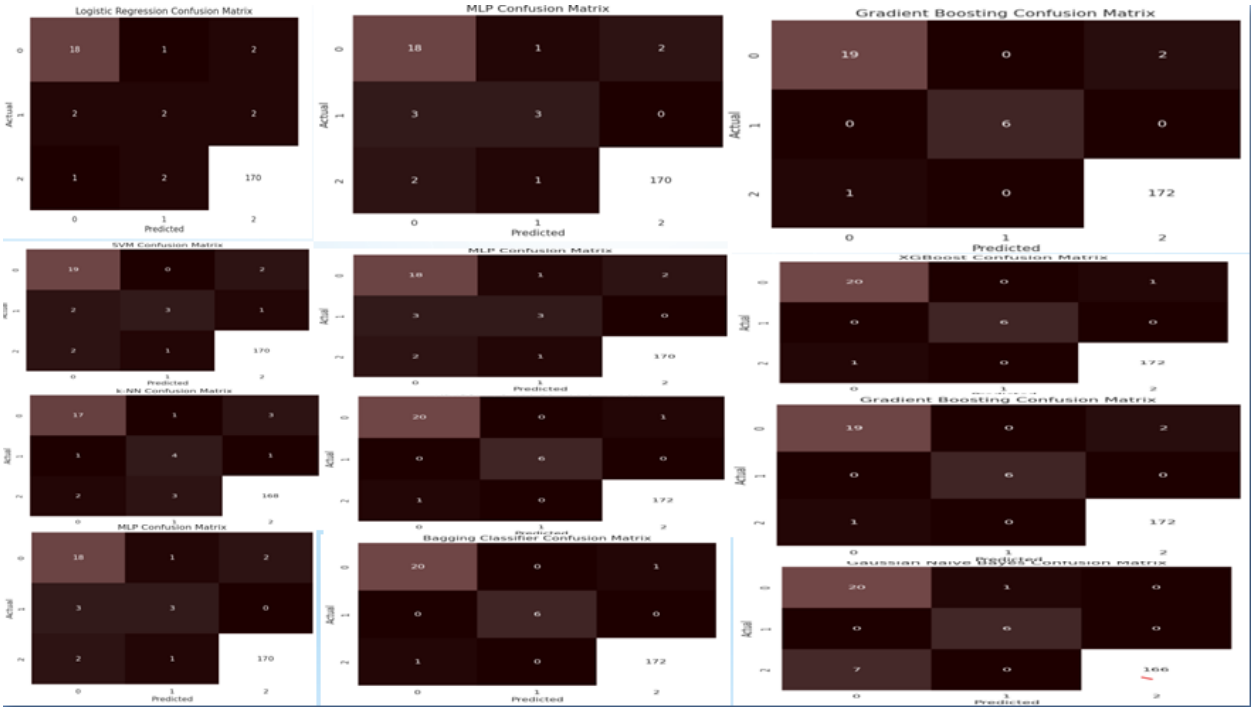

fig 44: output for all confusion matrix

Across the 12 models (Random Forest, SVM, Gradient Boosting, Decision Tree, Logistic Regression, k-NN, MLP, AdaBoost, Bagging, Voting, Extra Trees, XGBoost), consistent performance is observed in correctly identifying instances of Class 0, with TP counts ranging from 17 to 20. For Class 1, there's variability in TP counts (2 to 6), indicating differences in models' ability to predict this class. Class 2 exhibits stable TP counts (164 to 172) across all models, displaying consistent performance in identifying instances. Overall, models show effectiveness in distinguishing Class 0 and Class 2, with variability in performance for Class 1.

7.3 Separate roc for each class for best 10 models

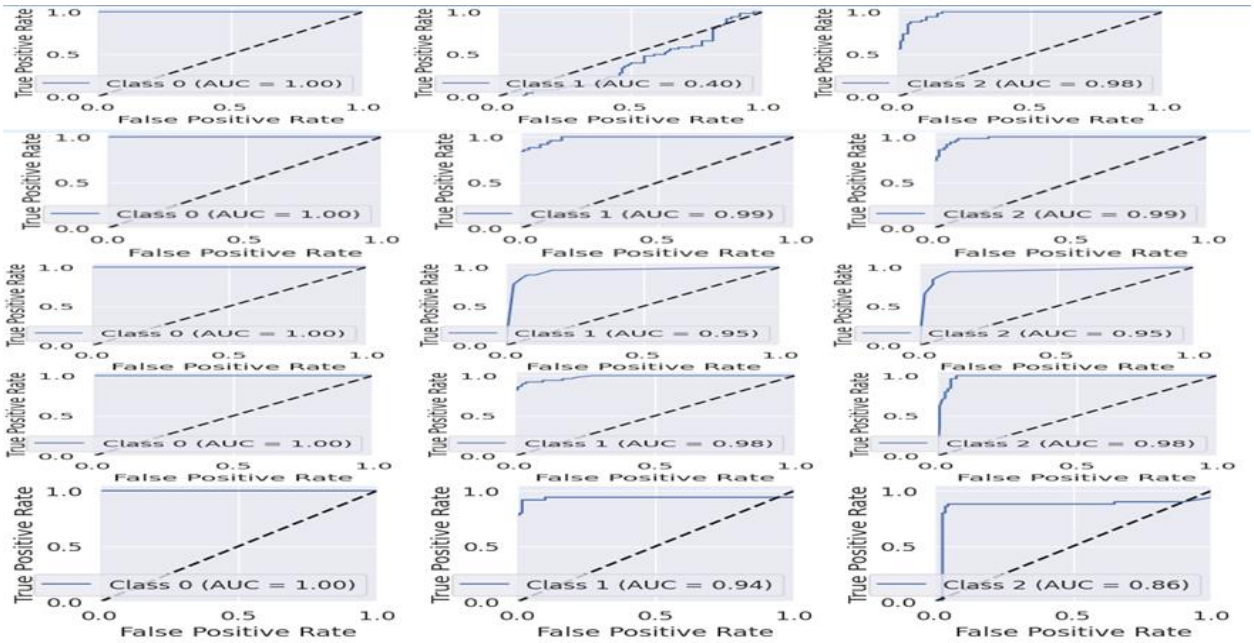

fig 45: Separate roc for each class for best models

Rocs for all class for all 10 models reveals class 0 predicted well

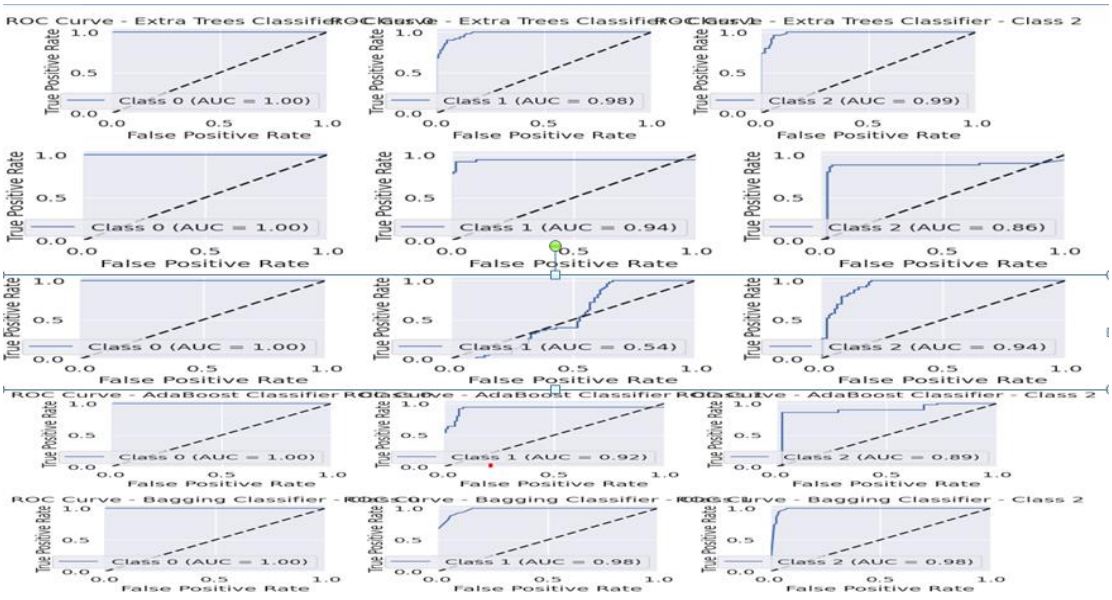

fig 46: Separate roc for each class for best models

Rocs for all class for all 10 models reveals class 0 predicted well

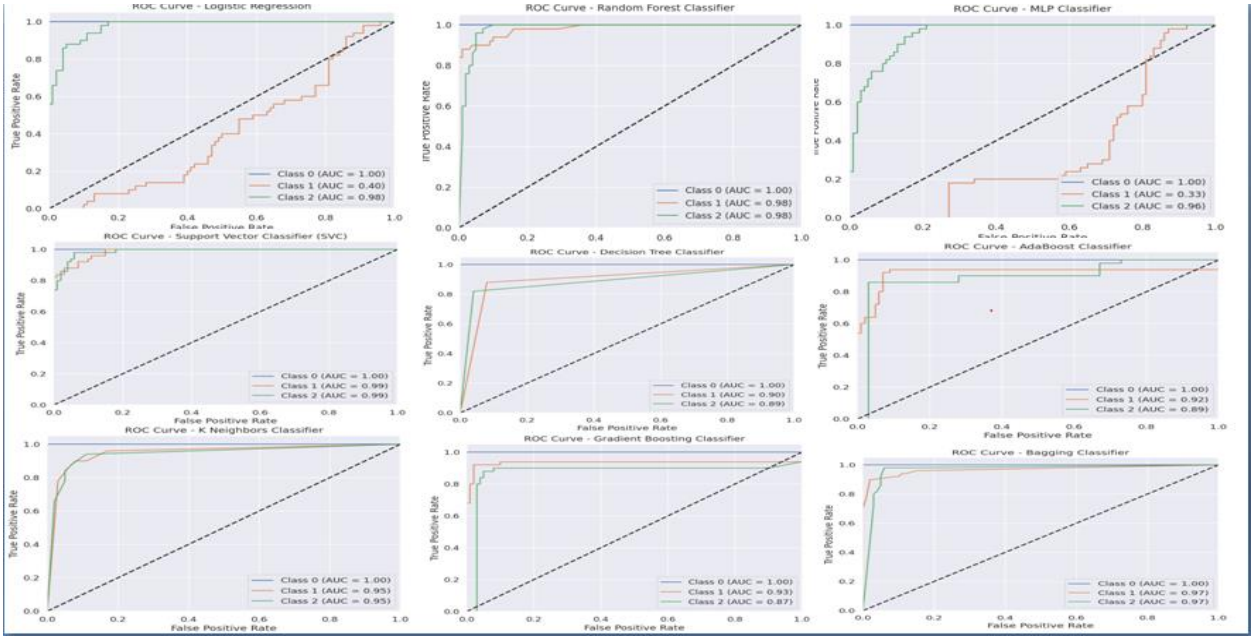

fig 47: roc for all class for models

Rocs for all class for all 10 models reveals class 0 predicted well

7.4 Combined ROC curves of all 10 models involving CLASS 0,1,2.

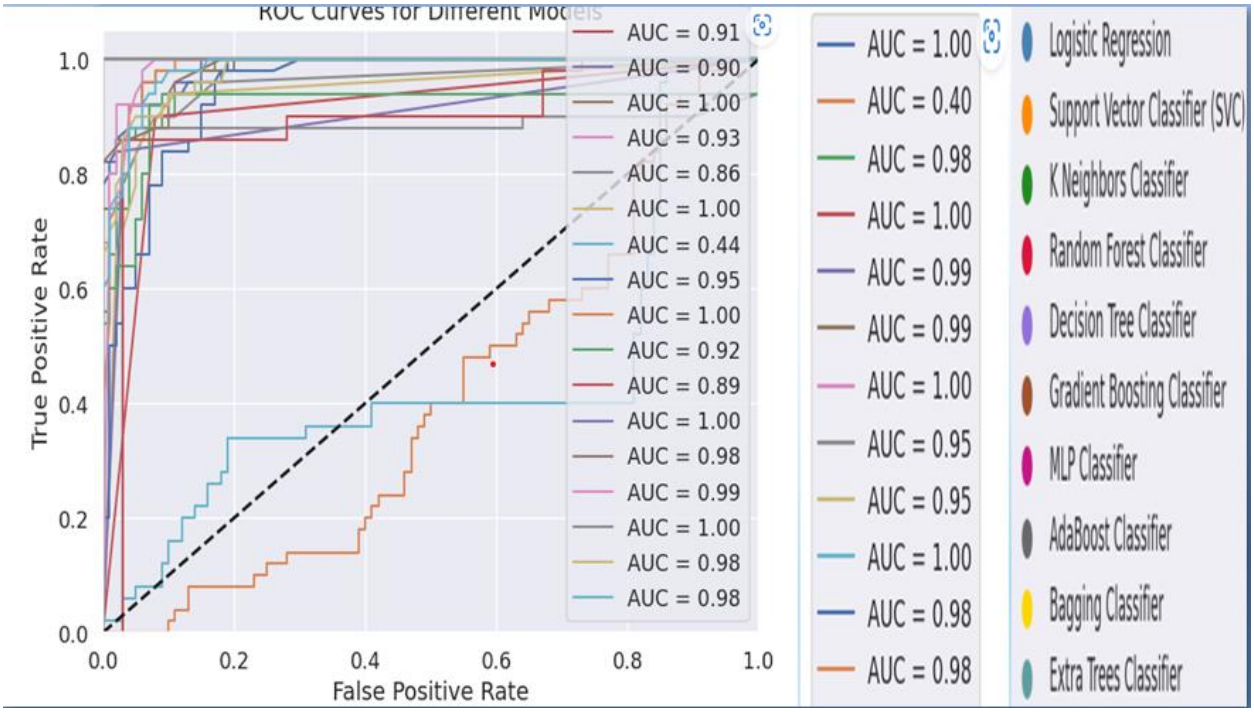

fig 48: Combined ROC curves of all 10 models involving CLASS 0,1,2.

The ROC curves illustrate the performance of diverse models (Logistic Regression, SVC, KNN, Random Forest, Decision Tree, Gradient Boosting, MLP, AdaBoost, Bagging, Extra Trees) across three classes. Class 0 shows impeccable discrimination (AUC=1) for all models. Class 1 exhibits variable performance (AUC=0.4 to 0.98), with notable high scores for SVC, Decision Tree, MLP, and Extra Trees (0.98, 0.9, 0.93, 0.95). Class 2 consistently performs well (AUC=0.89 to 0.99).

7.5 F1 SCORE, SUPPORT, RECALL, PRECISION COMPARISON FOR ALL MODELS

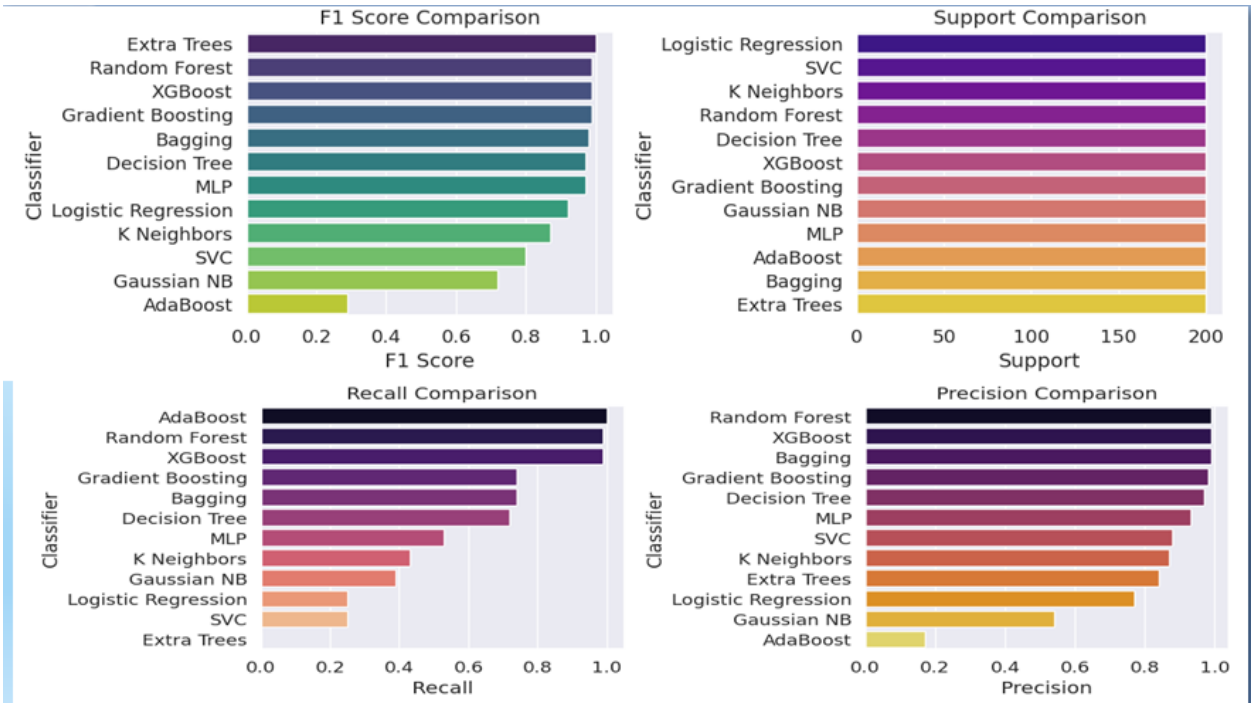

Fig 49: F1 SCORE, SUPPORT, RECALL, PRECISION COMPARISON FOR ALL MODELS

RandomForestClassifier and XGBClassifier outperformed other models with consistently high precision, recall, and F1-score across all classes. Logistic Regression and Support Vector Classifier (SVC) also performed well, though with slightly lower scores in some metrics, suggesting the need for more improvement.

7.6 OVERALL MODEL PERFORMANCE AND PRECISION COMPARISON

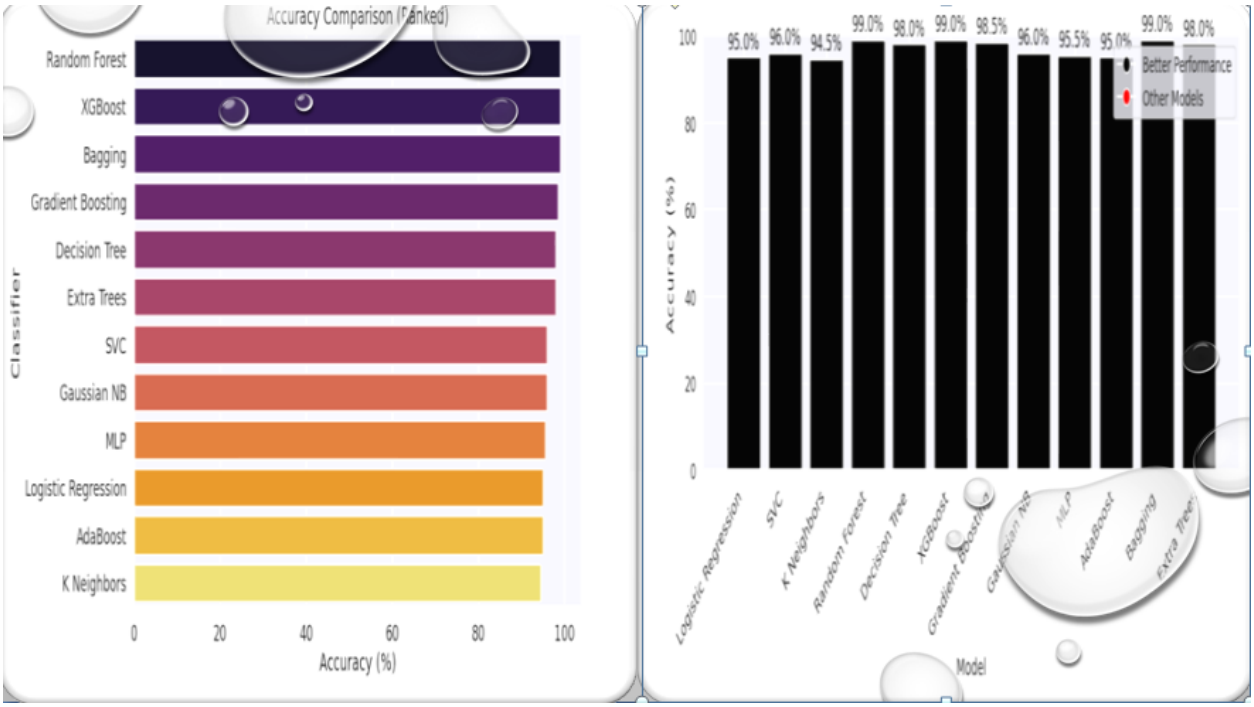

Fig 50: Overall model performance and precision comparison

In our evaluation of twelve diverse machine learning models, encompassing DecisionTreeClassifier, Logistic Regression, SupportVectorClassifier, KNeighborsClassifier, RandomForestClassifier, XGBClassifier, Neural Network, Naive Bayes, AdaBoostClassifier, GradientBoostingClassifier, SVM with RBF Kernel, and RandomizedForestClassifier, we assessed their performance using key evaluation metrics. These metrics included Accuracy, Precision, Recall, F1 Score, and ROC-AUC. Notably, the RandomForestClassifier emerged as the top-performing model, boasting the highest accuracy among all candidates. It exhibited exceptional precision across all classes, maintained consistently high recall values, indicating proficiency in capturing positive instances, and demonstrated a balanced F1 Score with elevated values. The ROC-AUC analysis underscored its robust discrimination ability. The XGBClassifier closely followed, securing the second-highest accuracy, with competitively high precision values, strong recall scores, balanced F1 performance, and a noteworthy ROC-AUC indicating strong discriminative capability. While these two models outperformed others, some alternative models displayed competitive results in specific metrics, necessitating a nuanced consideration of the trade-offs and characteristics of each model for informed decision-making in specific use cases.

8. Summary of the findings:

In our diabetes prediction project, a thorough analysis revealed notable findings. Following data preprocessing with Pandas, the dataset, comprising 1000 entries and 12 columns, provided valuable insights into diabetes-related variables. The 'CLASS' variable distribution displayed 844 instances of 'Y,' 102 of 'N,' and 53 of 'P.' Summary statistics unveiled standardized data, while normality tests, including Shapiro-Wilk and Anderson-Darling, indicated non-normal distribution across all variables, with

additional skewness identified in some. Association analyses demonstrated a moderate correlation (Cramér's V = 0.4133) between gender and diabetes status, further supported by a significant association (p-value < 0.05). Feature engineering and model optimization involved excluding less significant variables and applying Synthetic Minority Over-sampling Technique (SMOTE) for class balance. Machine learning model evaluation highlighted Random Forest, XGBoost, and Bagging as top performers, achieving an impressive 99.00% accuracy in multi-class classification. Notably, RandomForestClassifier emerged as the overall best model, exhibiting the highest accuracy, precision, recall, and balanced F1 score. These findings underscore the efficacy of machine learning models in diabetes prediction, shedding light on variable associations and emphasizing the importance of thoughtful feature selection for optimal model performance.

9.limitations:

9.1 Regional Specificity of the Dataset:

The dataset utilized in this study was sourced exclusively from a specific region in Iraq. While the findings are valuable within the context of this region, it is important to acknowledge the limitation that the generalizability of the results to broader populations may be constrained. Extrapolating these findings to other diverse populations requires caution and further research incorporating a more geographically diverse dataset.

9.2 Class Imbalance Despite SMOTE Application:

Despite the application of Synthetic Minority Over-sampling Technique (SMOTE) for Support Vector Classification (SVC) and Gradient Boosting, a notable class imbalance, especially concerning class 1, persists in the dataset. This imbalance may impact the model's performance, and additional fine-tuning measures are warranted. Further research and parameter tuning are recommended to address and mitigate the challenges posed by this imbalance, ensuring a more robust and equitable performance across different classes.

10.Results

We reject the null hypothesis since all the variable listed in research question has influence in predicting the status of diabetes

APPENDIX

11.References

Rashid, A. (2020). Diabetes Dataset. Data.mendeley.com, 1. https://doi.org/10.17632/wj9rwkp9c2.1

American Diabetes Association. (2021). Classification and diagnosis of diabetes: Standards of medical care in diabetes—2021. Diabetes Care, 44(Supplement 1), S15–S33. https://doi.org/10.2337/dc21-s002
Phongying, M., & Hiriote, S. (2023). Diabetes Classification Using Machine Learning Techniques. *Computation*, *11*(5), 96. https://doi.org/10.3390/computation11050096
Butt, U. M., Letchmunan, S., Ali, M., Hassan, F. H., Baqir, A., & Sherazi, H. H. R. (2021). Machine Learning Based Diabetes Classification and Prediction for Healthcare Applications. *Journal of Healthcare Engineering*, *2021*, 1–17. https://doi.org/10.1155/2021/9930985
Biswal, A. (2021, July 22). *What is Bagging in Machine Learning And How to Perform Bagging*. Simplilearn.com. https://www.simplilearn.com/tutorials/machine-learning-tutorial/bagging-in-machine-learning
Kanade, V. (2022). *What Is Logistic Regression? Equation, Assumptions, Types, and Best Practices*. Spiceworks. https://www.spiceworks.com/tech/artificial-intelligence/articles/what-is-logistic-regression/
Kanade, V. (2022, September 29). *What Is a Support Vector Machine? Working, Types, and Examples*. Spiceworks. https://www.spiceworks.com/tech/big-data/articles/what-is-support-vector-machine/
Zhang, Z. (2016). Introduction to machine learning: k-nearest neighbors. *Annals of Translational Medicine*, *4*(11), 218–218. https://doi.org/10.21037/atm.2016.03.37
Schonlau, M., & Zou, R. Y. (2020). The random forest algorithm for statistical learning. *The Stata Journal: Promoting Communications on Statistics and Stata*, *20*(1), 3–29. https://doi.org/10.1177/1536867x20909688
Song, Y.-Y., & Lu, Y. (2015). Decision tree methods: applications for classification and prediction. *Shanghai Archives of Psychiatry*, *27*(2), 130–135. https://doi.org/10.11919/j.issn.1002-0829.215044
Aliyev, V. (2020, October 7). *Gradient Boosting Classification explained through Python*. Medium. https://towardsdatascience.com/gradient-boosting-classification-explained-through-python-60cc980eeb3dRamakrishna, M. T., Venkatesan, V. K., Izonin, I., Havryliuk, M., & Bhat, C. R. (2023). Homogeneous Adaboost Ensemble Machine Learning Algorithms with Reduced Entropy on Balanced Data. *Entropy*, *25*(2), 245. https://doi.org/10.3390/e25020245
Confusion Matrix - an overview | ScienceDirect Topics. (2019). Sciencedirect.com. https://www.sciencedirect.com/topics/engineering/confusion-matrix
Bhandari, A. (2020, June 15). *AUC-ROC Curve in Machine Learning Clearly Explained*. Analytics Vidhya. https://www.analyticsvidhya.com/blog/2020/06/auc-roc-curve-machine-learning/
Karlo Abnoosian, Rahman Farnoosh, & Mohammad Hassan Behzadi. (2023). Prediction of diabetes disease using an ensemble of machine learning multi-classifier models. BMC Bioinformatics, 24(1). https://doi.org/10.1186/s12859-023-05465-z
Saeedi, P., Petersohn, I., Salpea, P., Malanda, B., Karuranga, S., Unwin, N., Colagiuri, S., Guariguata, L., Motala, A. A., Ogurtsova, K., Shaw, J. E., Bright, D., & Williams, R. (2019). Global and Regional Diabetes Prevalence Estimates for 2019 and Projections for 2030 and 2045: Results from the International Diabetes Federation Diabetes Atlas, 9th Edition. Diabetes Research and Clinical Practice, 157(157), 107843. https://doi.org/10.1016/j.diabres.2019.107843
W3Schools. (2023). *W3Schools Online Web Tutorials*. W3schools.com; W3Schools. https://www.w3schools.com/