

Programming for Bioinformatics (SECB3203)

Title: **Diabetic prediction using the machine learning**

PROGRESS 4

(CHAPTER 4)

Group 18

| Group Members | Matric Number |
|------------------------|---------------|
| HARCHANA A/P ARULAPPAN | A21EC0028 |
| MALAVIKA A/P BASKARAN | B22EC0069 |

TABLE OF CONTENTS

| | TITLE | PAGE |
|------------------|-------------------------------------|-------------|
| CHAPTER 1 | INTRODUCTION | |
| 1.0 | Introduction | 3 |
| 1.1 | Problem Background | 4 |
| 1.2 | Problem Statement | 5 |
| 1.3 | Aim | 6 |
| 1.4 | Objectives | 6 |
| 1.5 | Scopes | 6 |
| 1.6 | Summary | 7 |
| CHAPTER 2 | RESEARCH METHODOLOGY | |
| 2.0 | Software and Hardware Requirement | 8 |
| 2.1 | Flowchart of the proposed approach | 9 |
| CHAPTER 3 | DATA PREPROCESSING | |
| 3.0 | Importing Dataset | 13 |
| 3.1 | Check for Missing Values | 15 |
| 3.2 | Check missing values in each column | 16 |
| 3.3 | Check for duplicate rows | 17 |
| 3.4 | Data Formatting | 18 |
| 3.5 | Data Normalization | 19 |
| 3.6 | Data Binning | 20 |
| 3.7 | Summary | 21 |
| CHAPTER 4 | MODEL DEVELOPMENT | |
| 4.0 | Model Development Flowchart | 23 |
| 4.1 | Model Training | 24 |
| 4.2 | Prediction and Decision Making | 28 |
| 4.3 | Summary | 30 |

CHAPTER 1: INTRODUCTION

1.0 Introduction

Diabetes is a chronic health condition characterized by the body's inability to effectively utilize and regulate glucose, a vital source of energy for daily activities. This disease manifests in two primary forms: Type 1 diabetes, often afflicting children and young individuals, occurs when the body fails to produce the hormone insulin. In contrast, Type 2 diabetes predominantly affects adults, where the body struggles to produce sufficient insulin, resulting in elevated blood sugar levels. Detecting diabetes in its early stages is paramount for reducing the associated health risks and managing treatment costs.

The role of insulin, produced by the pancreas, is critical in regulating blood glucose levels. Several factors, including excessive body weight, sedentary lifestyles, high blood pressure, and abnormal cholesterol levels, can predispose individuals to diabetes. One common symptom of diabetes is increased urination. Left unmanaged, this condition can lead to various complications, such as skin, nerve, and eye damage, and in more severe cases, it may result in kidney failure and diabetic retinopathy, an ocular disease.

Understanding the significance of early detection and effective management of diabetes is essential to mitigate its potentially severe consequences and reduce the burden of treatment costs on individuals and healthcare systems. In this research, we will assess the effectiveness of three distinct classification techniques: Support Vector Machine (SVM), Random Forest, and K-Nearest Neighbors (K-NN), to classify the dataset based on the presence of diabetic disease, differentiating between normal and abnormal cases.

1.1 Problem Background

Diabetes is not merely a statistic but a pervasive health concern that significantly impacts the well-being of individuals. (Tasin et al., 2023). This chronic condition has the potential to induce severe health problems, including kidney failure, blindness, and more. Unfortunately, there is currently no cure for diabetes, making it imperative for patients to adopt a healthy lifestyle and meticulously manage their insulin intake.

Enter data mining, a technology that has gained significant traction in the healthcare industry. It presents an opportunity to reduce costs and improve the efficiency of diabetic disease detection by leveraging patient datasets, including variables such as age and number of pregnancies. In this research, we concentrate on a specific data mining method known as classification, aiming to explore and understand its potential in predicting diabetic disease.

The problem of predicting diabetic disease using machine learning techniques carries substantial significance in the field of healthcare and medical research. Diabetes is a prevalent, chronic ailment that affects millions of individuals across the globe. Timely and accurate detection, along with effective management, is essential to minimize the health-related consequences of this condition and alleviate the financial strain on healthcare systems and individuals.

The ability of machine learning to analyze vast datasets offers the promise of early intervention and personalized treatment plans for diabetes, thereby enhancing patient outcomes and quality of life. Furthermore, it facilitates resource allocation in healthcare, allowing for the identification of high-risk individuals who may require more intensive monitoring and preventive measures.

Given the intricate nature of diabetes and its multifaceted variables, the integration of cutting-edge machine learning algorithms, such as Support Vector Machine (SVM), Random

Forest, and K-Nearest Neighbors (K-NN), presents an innovative approach to addressing this public health challenge. This research aims to evaluate the performance of these machine learning techniques in predicting diabetic disease, ultimately contributing to the advancement of healthcare decision-making and improving the overall well-being of those impacted by this widespread and life-altering condition.

1.2 Problem Statement

Diabetes is a widespread and chronic health condition that affects millions of individuals globally, and it has a significant impact on the well-being of those afflicted. Early detection and effective management of diabetes are critical to mitigate its adverse health effects and reduce healthcare costs. Traditional clinical methods for diagnosing diabetes are often expensive and can place a financial burden on both healthcare systems and individuals, particularly those who may struggle to afford treatment.

Given the availability of extensive patient data, including variables such as age, lifestyle factors, and medical history, data mining techniques present an opportunity to improve the efficiency of diabetic disease detection and prediction. This research aims to address the following problem:

Can machine learning, specifically the application of classification algorithms such as Support Vector Machine (SVM), Random Forest, and K-Nearest Neighbors (K-NN), enhance the accuracy and cost-effectiveness of predicting diabetic disease based on patient data, ultimately improving patient outcomes and healthcare resource allocation?

This problem statement encapsulates the overarching challenge of utilizing advanced machine learning methods to develop accurate predictive models for diabetes, with the potential to enable early intervention, personalized treatment plans, and cost savings within healthcare systems.

1.3 Aim

The aim of this project is to design and develop classification of diabetes prediction using three different algorithms which is Support Vector Machine(SVM), Random Forest and K-Nearest Neighbors (K-NN).

1.4 Objectives

The objective of this project are:

1. To study the dataset, classification methods, and diabetes domain to determine the domain, methods, and dataset used in Support Vector Machine(SVM), Random Forest and K-Nearest Neighbors (K-NN).
2. To analyze and predict the dataset of diabetes disease by using three different algorithm Support Vector Machine(SVM), Random Forest and K-Nearest Neighbors (K-NN) in order to determine which dataset are in the normal or abnormal presence of the diabetes disease.
3. To evaluate the performance of Support Vector Machine(SVM), Random Forest, and K-Nearest Neighbors(K-NN) in terms of accuracy and precision.

1.5 Scopes

The scope of this project are:

1. Using Support Vector Machine(SVM), Random Forest and K-Nearest Neighbour(K-NN) techniques for diabetes classification.
2. Using the dataset from UCI Pima Indians Diabetes.
3. Using the performance measurement classification such as early detection, accuracy and precision.

1.6 Summary

In conclusion, diabetes is a chronic health problem and it can be a reason to lower the lifespan as well as quality of life. By developing a Diabetes Prediction using Machine Learning we can predict the risk of diabetes at an early stage of an individual. This may help them to take action on their health care and they can arrange their diet plan accordingly. For this project we will use three different algorithm for classification method in identifying normal and abnormal cases in the dataset based on the presence of diabetes. The algorithm we will use for this project is Support Vector Machine(SVM), Random Forest and K-Nearest Neighbors (K-NN).

CHAPTER 2 : METHODOLOGY

2.0 Software and Hardware Requirements

Hardware Requirements

1. Laptop :

- a. For the purpose of handling data preparation, model training, and evaluation, laptops are required with enough processing power and memory. The specific requirements will depend on the size of the dataset and the complexity of the models we're working with.

2. Storage :

- a. Storage space needed to store the diabetic data, any intermediate files, and the trained models. Ensure you have enough disk space for your dataset and associated files.

Software Requirements

1. Data Analysis and Preprocessing:

- a. Python: Python and bio-python is a popular language for data analysis and machine learning.

2. Machine Learning Libraries:

- a. Scikit-Learn: This library provides tools for data preprocessing, model building, and evaluation. It supports SVM, Random Forest, and K-NN.
- b. LS-SVM: Required to install specific packages or libraries for implementing LS-SVM where it can be found LS-SVM packages in Python.

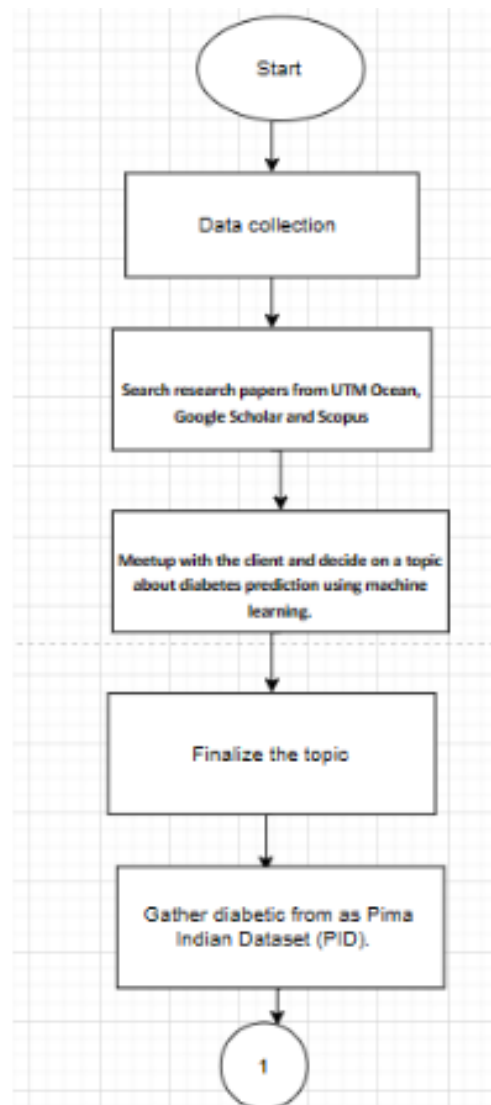
3. Operating System:

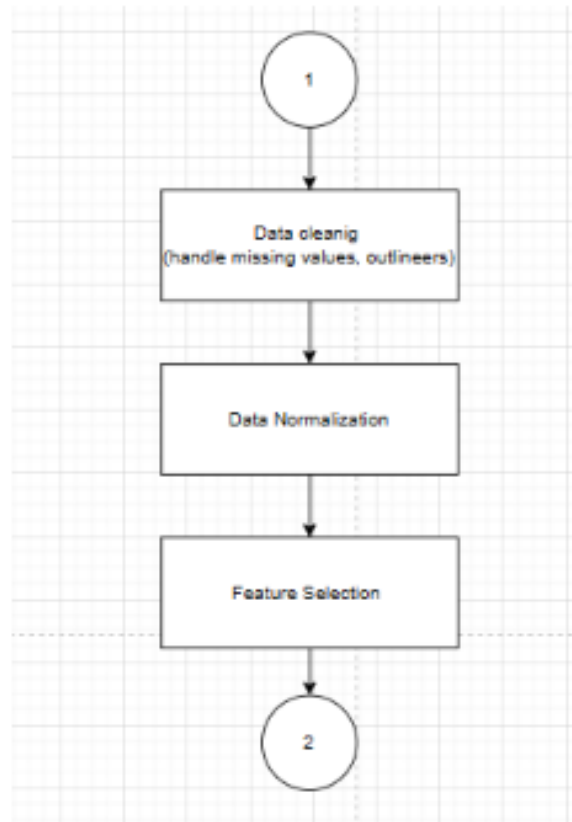
- a. Linux and Windows

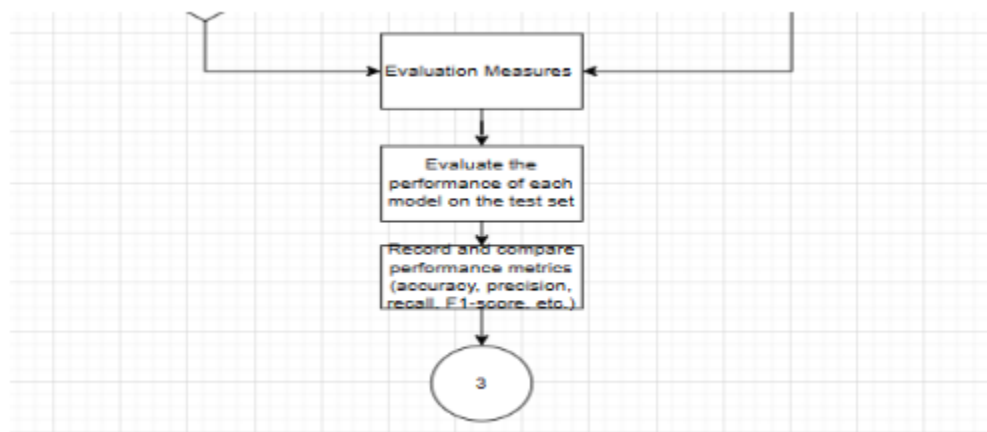
4. Data Visualization

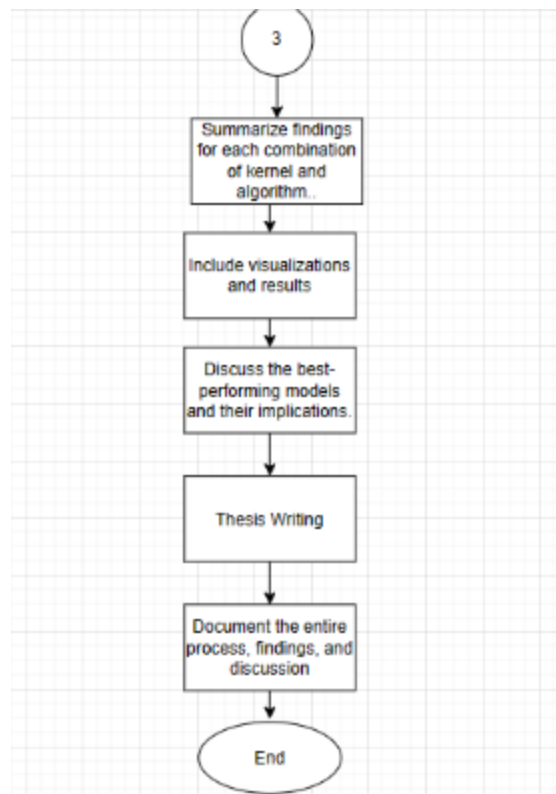
- a. Orange : For creating data visualizations and plots.

2.1 Flowchart









CHAPTER 3 : DATA PREPROCESSING

3.0 Importing Dataset

3.0.1 Understanding the data

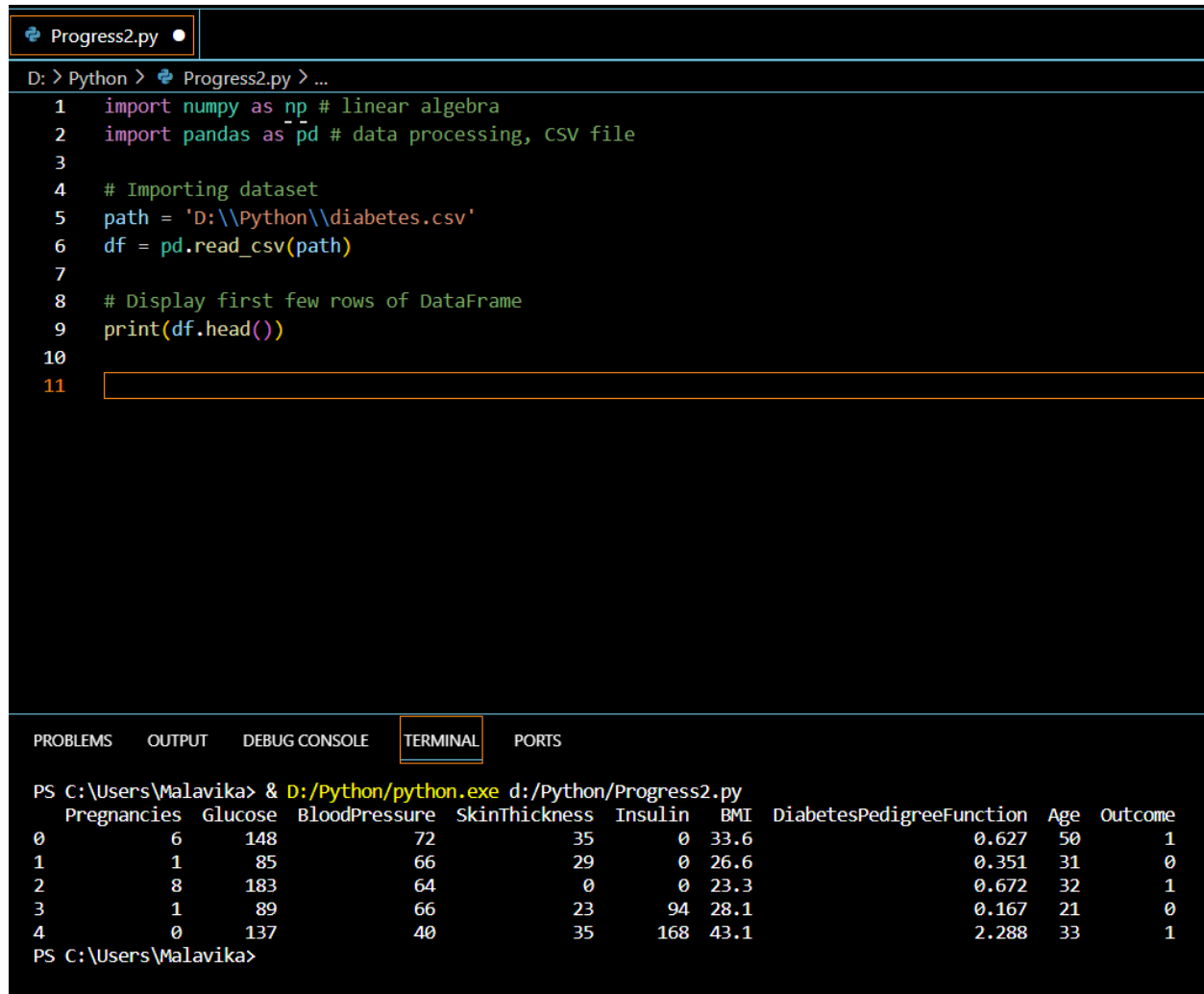
In this research, a dataset is being utilized to predict diabetes among pregnant women. This dataset incorporates various factors including the month of pregnancy, glucose levels, blood pressure, skin thickness, insulin, BMI, diabetic pedigree function, and age. The dataset originates from Kaggle and represents the Pima Indians, comprising information from 768 pregnant women. With 8 features available in the dataset, each feature plays a crucial role in understanding and predicting the likelihood of diabetes among pregnant women.

Our dataset from kaggle : <https://www.kaggle.com/datasets/uciml/pima-indians-diabetes-database>

| Features | Explanation |
|----------------------------|---|
| Month of Pregnancy | Could indicate varying stages, affecting how diabetes risk changes during pregnancy. |
| Glucose Levels | High levels may strongly correlate with the presence or risk of diabetes. |
| Blood Pressure | Elevated levels can be an indicator or risk factor for diabetes. |
| Skin Thickness | May have correlations with insulin resistance or other diabetic indicators. |
| Insulin Levels | A significant marker; irregular levels are often associated with diabetes. |
| BMI (Body Mass Index) | Higher BMI or obesity is a known risk factor for diabetes. |
| Diabetic Pedigree Function | Reflects genetic predisposition or familial history of diabetes. |
| Age | Age often correlates with increased diabetes risk due to various physiological changes over time. |

3.0.2 Importing and Exporting the data

We've imported the diabetes.csv file from the local directory in Python and display the result to the first 5 rows.



The screenshot shows an IDE with a Python script named `Progress2.py` and its output in the terminal. The script imports `numpy` and `pandas`, reads a CSV file from `D:\\Python\\diabetes.csv`, and displays the first five rows of the resulting DataFrame.

```
1 import numpy as np # linear algebra
2 import pandas as pd # data processing, CSV file
3
4 # Importing dataset
5 path = 'D:\\Python\\diabetes.csv'
6 df = pd.read_csv(path)
7
8 # Display first few rows of DataFrame
9 print(df.head())
10
11
```

The terminal output shows the command prompt running the script and displaying the first five rows of the DataFrame:

```
PS C:\Users\Malavika> & D:/Python/python.exe d:/Python/Progress2.py
Pregnancies  Glucose  BloodPressure  SkinThickness  Insulin  BMI  DiabetesPedigreeFunction  Age  Outcome
0           6      148             72           35         0   33.6              0.627   50         1
1           1       85             66           29         0   26.6              0.351   31         0
2           8      183             64            0         0   23.3              0.672   32         1
3           1       89             66           23        94   28.1              0.167   21         0
4           0      137             40           35       168   43.1              2.288   33         1
PS C:\Users\Malavika>
```

Figure 3.0 : Importing the raw data

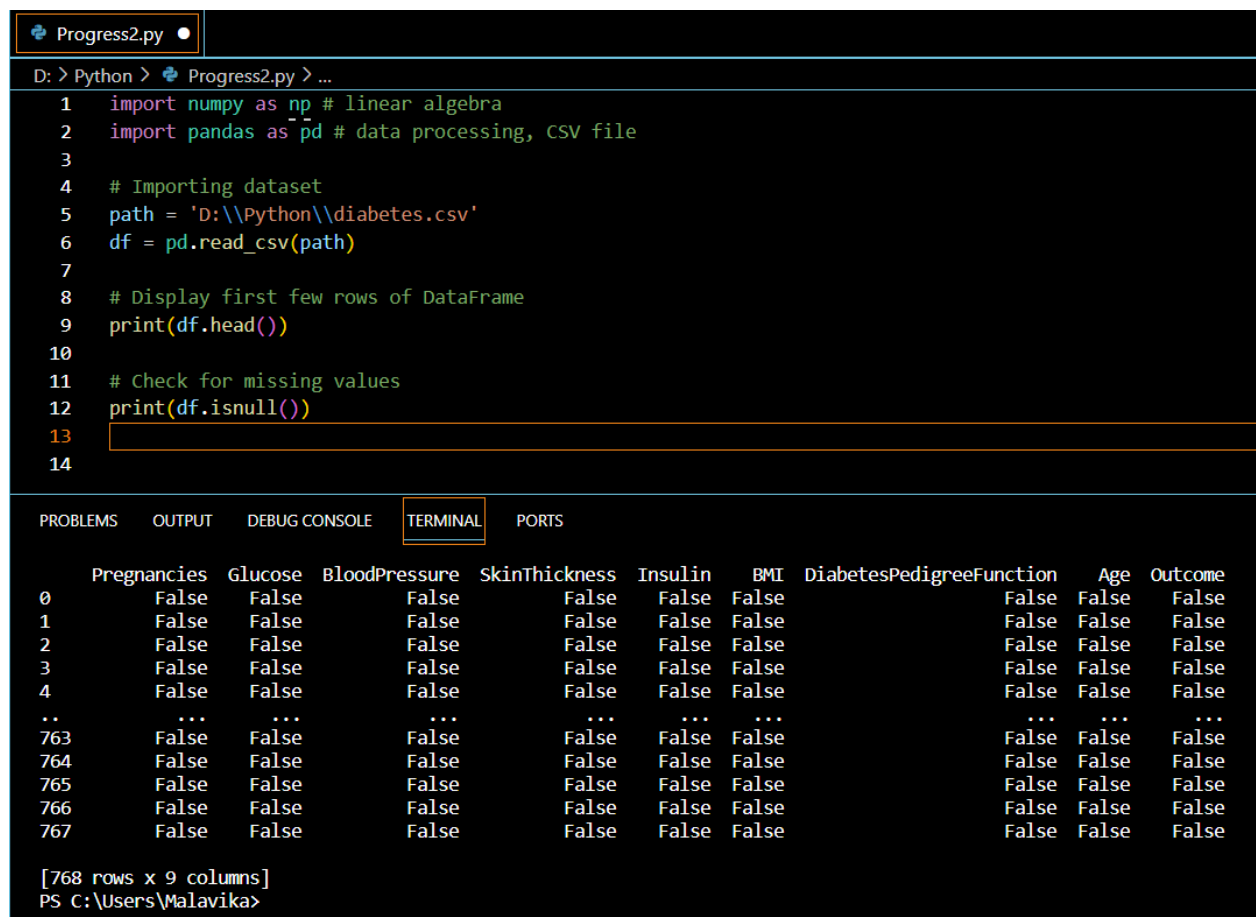
3.0.3 Getting Started Analyzing Data in Python

The data analysis is started with getting know the number of rows and columns the dataset has. Rows and columns of the dataset

3.1 Check for Missing Value

A comprehensive check for missing values was conducted. It's crucial to ensure the completeness of the dataset before any analysis. Upon thorough inspection, it was confirmed that our dataset exhibits no missing values.

This absence of missing data provides a strong foundation for accurate analysis and model building, allowing us to utilize the entirety of the dataset in our predictive modeling without the need for imputation or handling missing values. This ensures the robustness and reliability of our analyses, enabling us to derive meaningful insights and make accurate predictions regarding diabetes among pregnant women.



```
Progress2.py
D: > Python > Progress2.py > ...
1 import numpy as np # linear algebra
2 import pandas as pd # data processing, CSV file
3
4 # Importing dataset
5 path = 'D:\\Python\\diabetes.csv'
6 df = pd.read_csv(path)
7
8 # Display first few rows of DataFrame
9 print(df.head())
10
11 # Check for missing values
12 print(df.isnull())
13
14
```

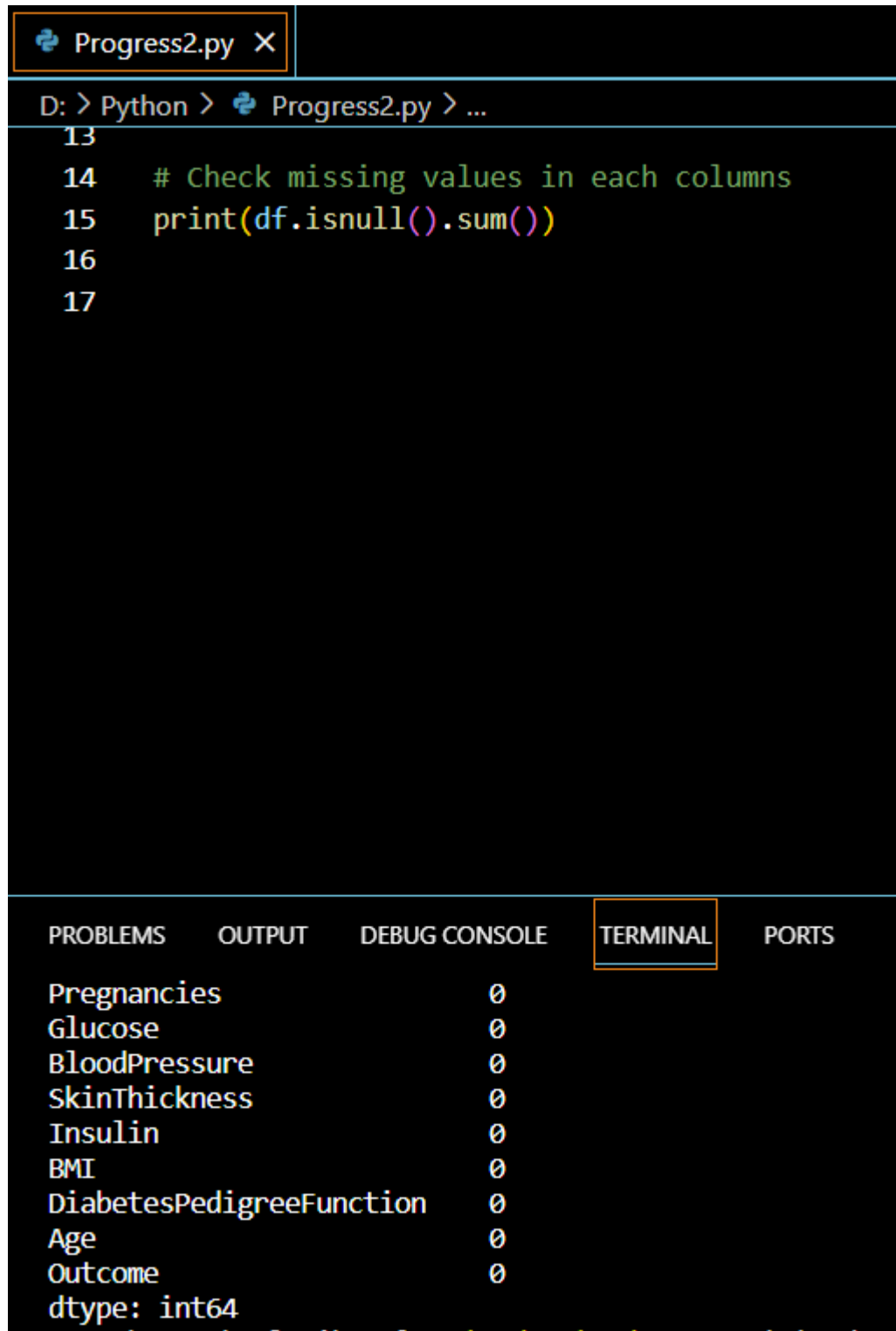
| | Pregnancies | Glucose | BloodPressure | SkinThickness | Insulin | BMI | DiabetesPedigreeFunction | Age | Outcome |
|-----|-------------|---------|---------------|---------------|---------|-------|--------------------------|-------|---------|
| 0 | False | False | False | False | False | False | False | False | False |
| 1 | False | False | False | False | False | False | False | False | False |
| 2 | False | False | False | False | False | False | False | False | False |
| 3 | False | False | False | False | False | False | False | False | False |
| 4 | False | False | False | False | False | False | False | False | False |
| .. | ... | ... | ... | ... | ... | ... | ... | ... | ... |
| 763 | False | False | False | False | False | False | False | False | False |
| 764 | False | False | False | False | False | False | False | False | False |
| 765 | False | False | False | False | False | False | False | False | False |
| 766 | False | False | False | False | False | False | False | False | False |
| 767 | False | False | False | False | False | False | False | False | False |

[768 rows x 9 columns]
PS C:\Users\Malavika>

Figure 3.1: No missing values

3.2 Check missing values in each column

Then, we checked missing values in each column.



The screenshot shows a Jupyter Notebook interface with a terminal window open. The terminal displays the execution of a Python script named `Progress2.py`. The script contains the following code:

```
13
14 # Check missing values in each columns
15 print(df.isnull().sum())
16
17
```

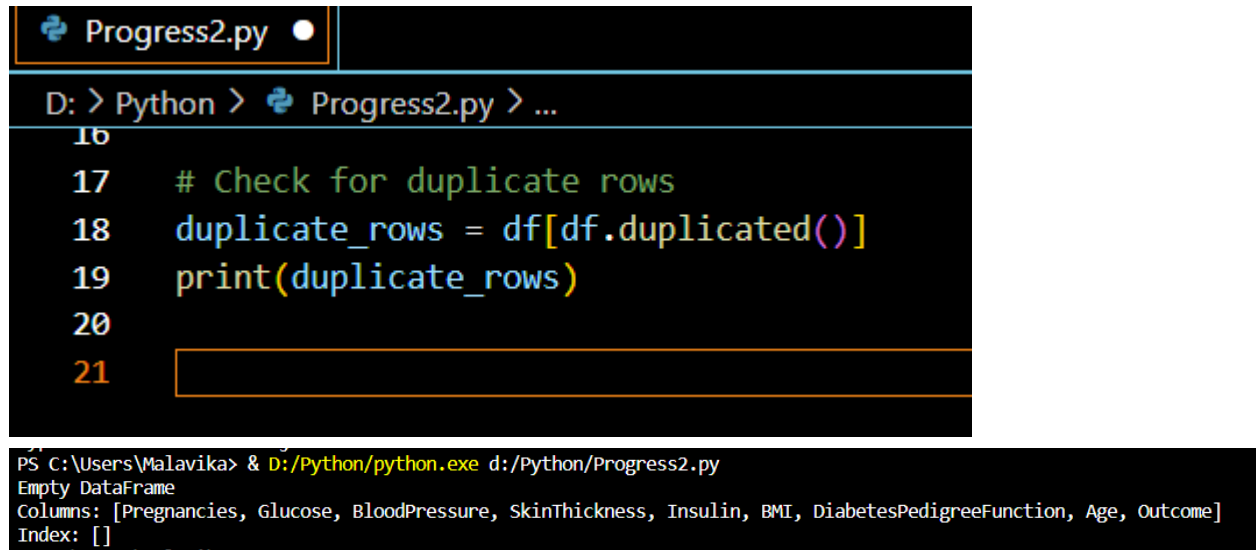
The output of the script is displayed in the terminal, showing the sum of missing values for each column:

| | PROBLEMS | OUTPUT | DEBUG CONSOLE | TERMINAL | PORTS |
|--------------------------|----------|--------|---------------|----------|-------|
| Pregnancies | | | 0 | | |
| Glucose | | | 0 | | |
| BloodPressure | | | 0 | | |
| SkinThickness | | | 0 | | |
| Insulin | | | 0 | | |
| BMI | | | 0 | | |
| DiabetesPedigreeFunction | | | 0 | | |
| Age | | | 0 | | |
| Outcome | | | 0 | | |
| dtype: | int64 | | | | |

Figure 3.2: No missing values in each column

3.3 Check for duplicate rows

Then, we checked for any duplicate rows in the dataset and we found that there are no duplicate rows.



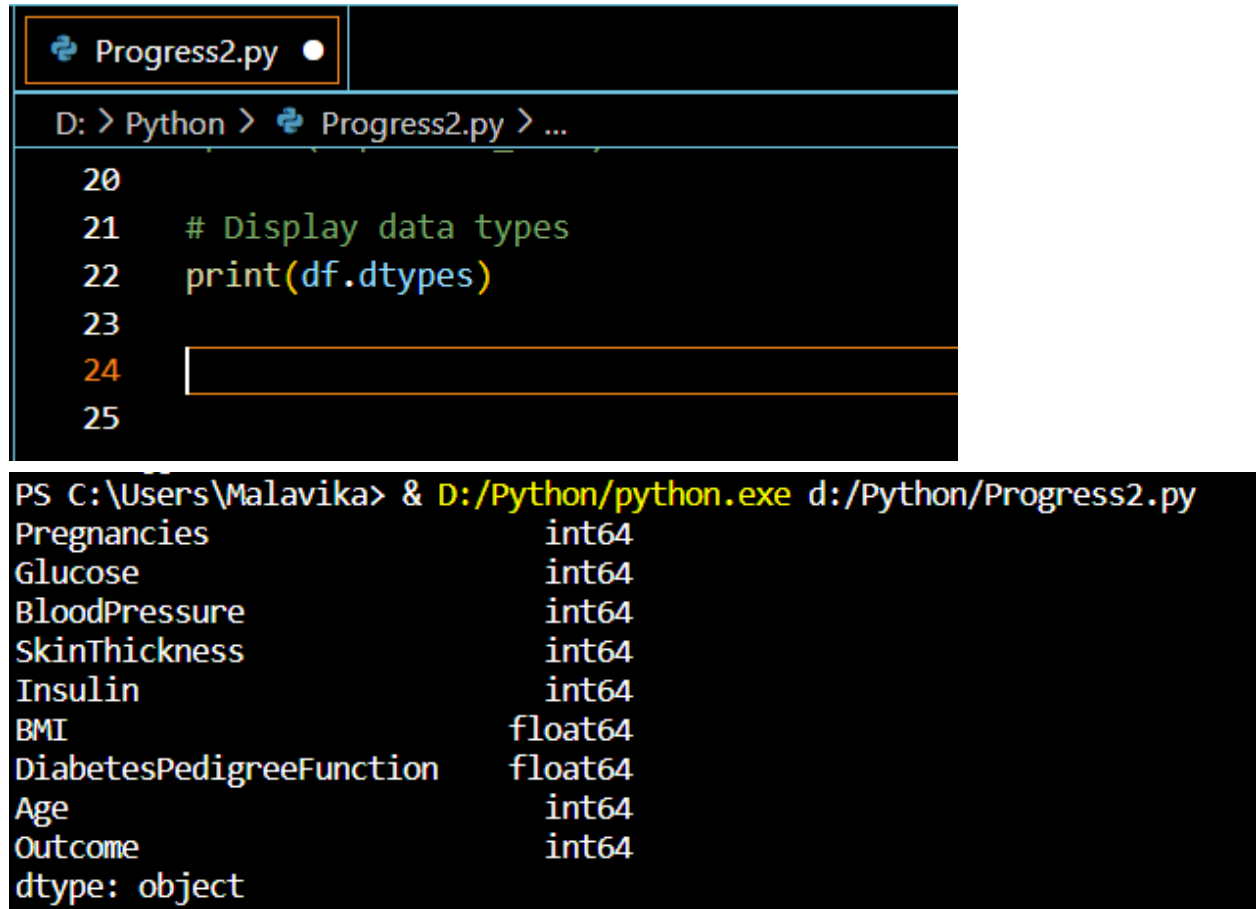
```
Progress2.py
D: > Python > Progress2.py > ...
16
17 # Check for duplicate rows
18 duplicate_rows = df[df.duplicated()]
19 print(duplicate_rows)
20
21

PS C:\Users\Malavika> & D:/Python/python.exe d:/Python/Progress2.py
Empty DataFrame
Columns: [Pregnancies, Glucose, BloodPressure, SkinThickness, Insulin, BMI, DiabetesPedigreeFunction, Age, Outcome]
Index: []
```

Figure 3.3 : No duplicate values

3.4 Data Formatting

The analysis is then proceeded with getting to know the data types of each columns



The image shows a code editor window titled 'Progress2.py' with the following code:

```
20
21 # Display data types
22 print(df.dtypes)
23
24
25
```

Below the code editor is a terminal window showing the command prompt and the output of the script:

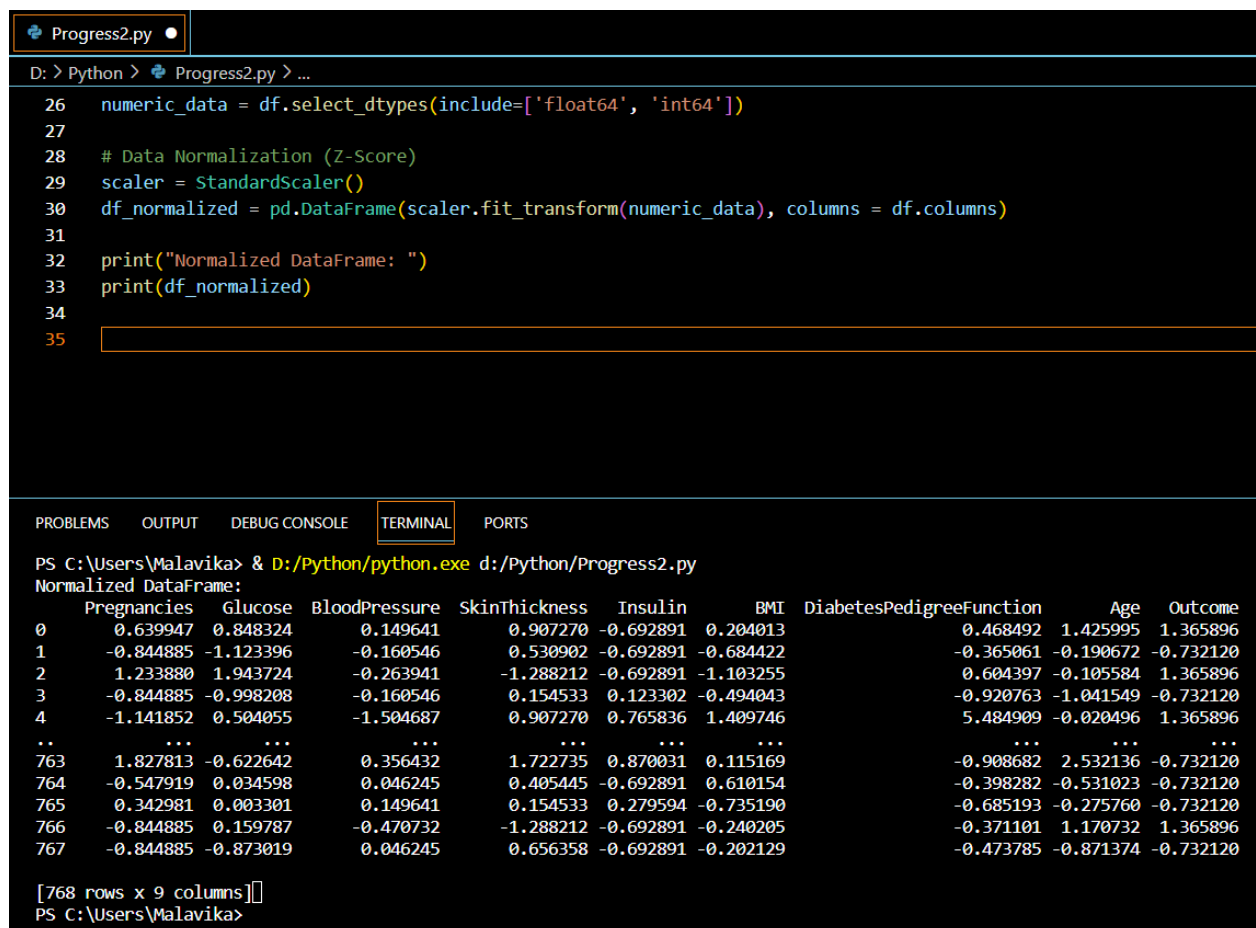
```
PS C:\Users\Malavika> & D:/Python/python.exe d:/Python/Progress2.py
Pregnancies      int64
Glucose          int64
BloodPressure    int64
SkinThickness    int64
Insulin          int64
BMI              float64
DiabetesPedigreeFunction float64
Age              int64
Outcome          int64
dtype: object
```

Figure 3.4 Display data types in each column.

3.5 Data Normalization

For data normalization, we use Z-score to measure how many standard deviations a data point is from the mean of a dataset. Data normalization is a process used to standardize the range of values in a dataset. The Z-score normalization method, also known as standardization, is a popular technique used in statistics and machine learning.

Z-score normalization involves centering the data around its mean. Subtracting the mean from each data point ensures that the resulting standardized values have a mean of zero. This centering process is a fundamental aspect of Z-score normalization, and it helps in making the data comparable and interpretable across different features or variables.



The screenshot shows a Jupyter Notebook interface with a file named 'Progress2.py'. The code in the notebook performs data normalization using Z-score. Below the code, the terminal output displays the resulting 'Normalized DataFrame' with 768 rows and 9 columns. The columns are: Pregnancies, Glucose, BloodPressure, SkinThickness, Insulin, BMI, DiabetesPedigreeFunction, Age, and Outcome. The values are centered around zero for most features, while Age and Outcome remain in their original scales.

```
26 numeric_data = df.select_dtypes(include=['float64', 'int64'])
27
28 # Data Normalization (Z-Score)
29 scaler = StandardScaler()
30 df_normalized = pd.DataFrame(scaler.fit_transform(numeric_data), columns = df.columns)
31
32 print("Normalized DataFrame: ")
33 print(df_normalized)
34
35
```

PROBLEMS OUTPUT DEBUG CONSOLE **TERMINAL** PORTS

```
PS C:\Users\Malavika> & D:/Python/python.exe d:/Python/Progress2.py
Normalized DataFrame:
   Pregnancies  Glucose  BloodPressure  SkinThickness  Insulin   BMI  DiabetesPedigreeFunction  Age  Outcome
0      0.639947  0.848324      0.149641      0.907270 -0.692891  0.204013      0.468492  1.425995  1.365896
1     -0.844885 -1.123396     -0.160546      0.530902 -0.692891 -0.684422     -0.365061 -0.190672 -0.732120
2      1.233880  1.943724     -0.263941     -1.288212 -0.692891 -1.103255      0.604397 -0.105584  1.365896
3     -0.844885 -0.998208     -0.160546      0.154533  0.123302 -0.494043     -0.920763 -1.041549 -0.732120
4     -1.141852  0.504055     -1.504687      0.907270  0.765836  1.409746      5.484909 -0.020496  1.365896
..         ...      ...      ...      ...      ...      ...      ...      ...      ...
763     1.827813 -0.622642      0.356432      1.722735  0.870031  0.115169     -0.908682  2.532136 -0.732120
764    -0.547919  0.034598      0.046245      0.405445 -0.692891  0.610154     -0.398282 -0.531023 -0.732120
765     0.342981  0.003301      0.149641      0.154533  0.279594 -0.735190     -0.685193 -0.275760 -0.732120
766    -0.844885  0.159787     -0.470732     -1.288212 -0.692891 -0.240205     -0.371101  1.170732  1.365896
767    -0.844885 -0.873019      0.046245      0.656358 -0.692891 -0.202129     -0.473785 -0.871374 -0.732120

[768 rows x 9 columns]
```

PS C:\Users\Malavika>

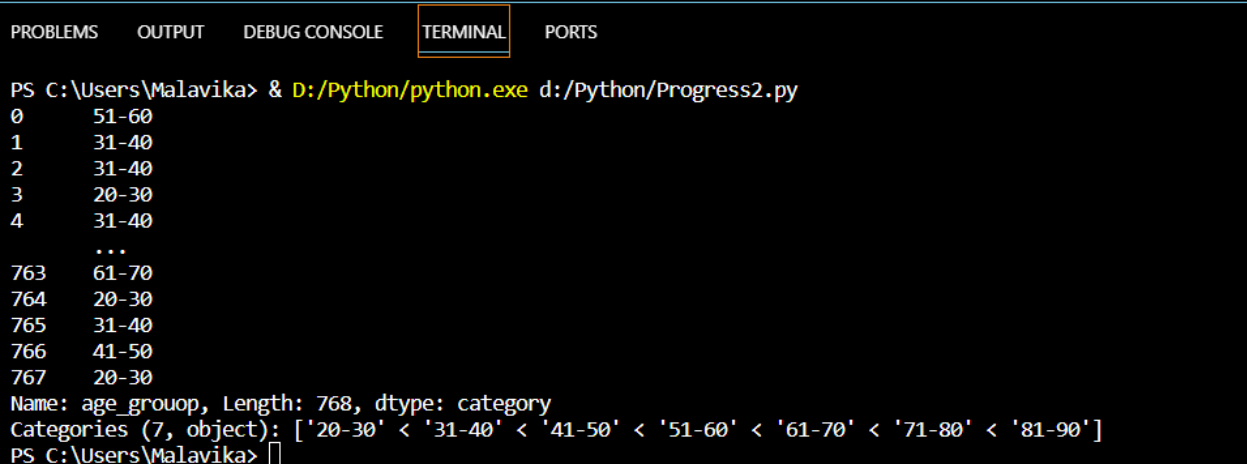
Figure 3.5 :Data Normalization: Centering output

3.6 Data Binning

We use age groups for data binning.

Data binning, also known as bucketing or discretization, is a technique used to group continuous numerical data into specific intervals or bins. When dealing with age groups, data binning involves categorizing ages into distinct ranges or groups for better analysis and interpretation.

```
35 # Binning the 'age' columns
36 bins = [20, 30, 40, 50, 60, 70, 80, 90]
37 labels = ['20-30', '31-40', '41-50', '51-60', '61-70', '71-80', '81-90']
38 df['age_grouop'] = pd.cut(df['Age'], bins=bins, labels=labels, right=False)
39 print(df['age_grouop'])
40
41
```



```
PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL PORTS

PS C:\Users\Malavika> & D:/Python/python.exe d:/Python/Progress2.py
0      51-60
1      31-40
2      31-40
3      20-30
4      31-40
...
763    61-70
764    20-30
765    31-40
766    41-50
767    20-30
Name: age_grouop, Length: 768, dtype: category
Categories (7, object): ['20-30' < '31-40' < '41-50' < '51-60' < '61-70' < '71-80' < '81-90']
PS C:\Users\Malavika>
```

Figure 3.6 :Data binning (Age group)

Summary of Chapter 3

This chapter covers fundamental steps in Python for preparing and analyzing data effectively in data science workflows. We start by stressing the importance of clean, well-structured data for meaningful insights. Exploring multiple data import methods using Pandas, we ensure smooth handling of various file formats. Moving to analysis, Python's tools like NumPy, Pandas, and Matplotlib are showcased for exploring, manipulating, and visualizing data. Real-life examples demonstrate turning raw data into actionable insights for better decision-making. The third section focuses on data wrangling, addressing missing values, optimizing data clarity, and normalizing techniques like centering and binning, all crucial for reliable analyses. This comprehensive understanding is reinforced with practical exercises and examples.

Chapter 4: Model Development

4.0 Machine Learning Model Development Flowchart

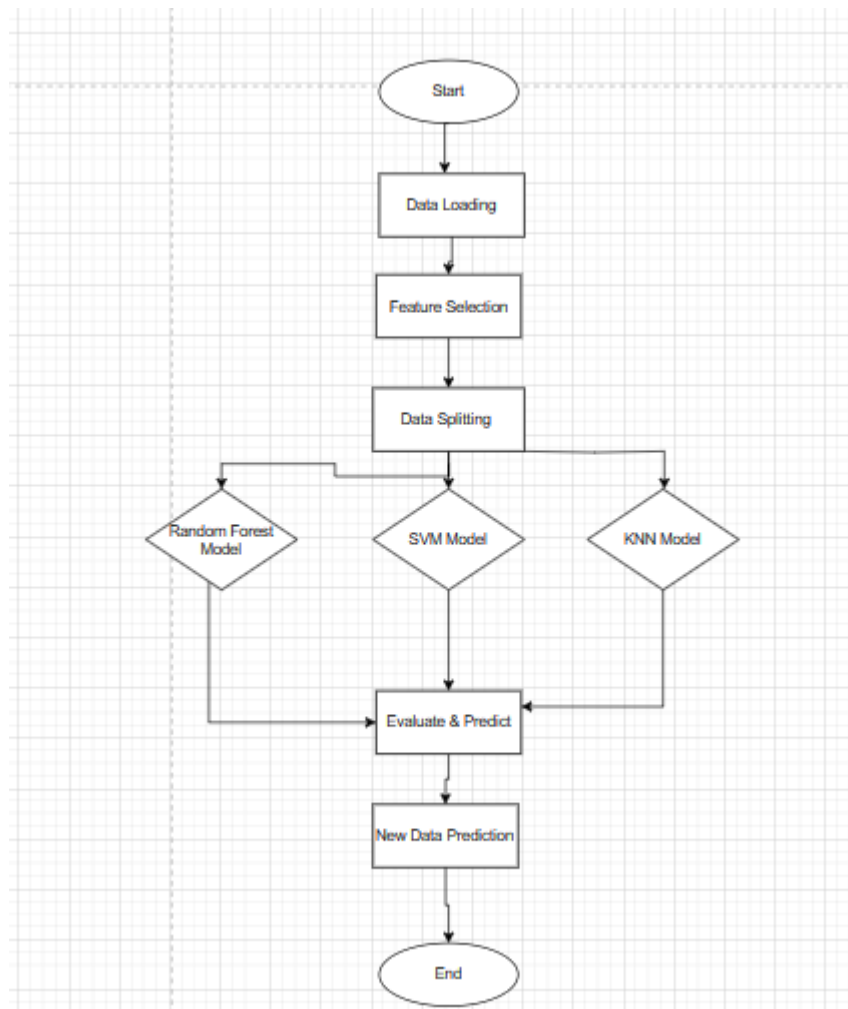


Figure 4.1

This flowchart illustrates the step-by-step process for developing machine learning models to predict diabetes. It begins with loading the dataset, selecting features, and splitting the data for training and testing. It then branches into training, evaluating, and predicting using Random Forest, Support Vector Machine (SVM), and K-Nearest Neighbors (KNN) models. Finally, it concludes with the prediction of outcomes for new data, offering a comprehensive overview of the model development workflow for diabetes prediction.

4.1 Model Training

Features and Target

```
# Selecting features and target
features = ['Pregnancies', 'Glucose', 'BloodPressure', 'SkinThickness', 'Insulin', 'BMI', 'DiabetesPedigreeFunction', 'Age']
X = df[features]
y = df['Outcome'] # 'Outcome' is a classification target variable
```

A) Random Forest:

| | | | | |
|---------------|------------------------------|--------|----------|---------|
| Random Forest | Accuracy: 0.7207792207792207 | | | |
| | precision | recall | f1-score | support |
| 0 | 0.79 | 0.78 | 0.78 | 99 |
| 1 | 0.61 | 0.62 | 0.61 | 55 |
| accuracy | | | 0.72 | 154 |
| macro avg | 0.70 | 0.70 | 0.70 | 154 |
| weighted avg | 0.72 | 0.72 | 0.72 | 154 |

Figure 4.2

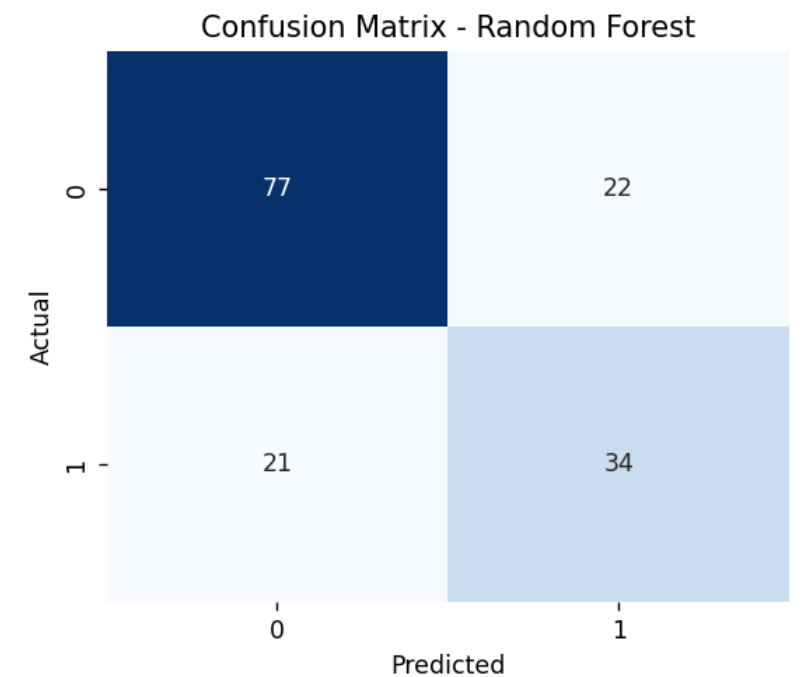


Figure 4.3

Based on the figure 4.2, an accuracy of 72.07% for the Random Forest model suggests that nearly 72 out of every 100 predictions made by the model align with the actual outcomes in the test dataset. This performance showcases the model's ability to generalize and make correct predictions. Random Forests are known for their ensemble nature, amalgamating multiple decision trees to offer robust predictions. Despite this respectable accuracy, there's room for improvement. Fine-tuning model parameters, such as the number of trees or their depth, might elevate the accuracy. Additionally, considering domain-specific features and understanding the clinical significance of correct predictions in diabetes diagnosis could further enhance the model's effectiveness. Evaluating other metrics and potentially addressing class imbalances, if present, could provide a more comprehensive assessment of the model's performance, ensuring its reliability in practical applications.

B) SVM:

| | | | | | |
|----------------------------------|-----------|--------|----------|---------|--|
| SVM Accuracy: 0.7662337662337663 | | | | | |
| | precision | recall | f1-score | support | |
| 0 | 0.78 | 0.88 | 0.83 | 99 | |
| 1 | 0.72 | 0.56 | 0.63 | 55 | |
| accuracy | | | 0.77 | 154 | |
| macro avg | 0.75 | 0.72 | 0.73 | 154 | |
| weighted avg | 0.76 | 0.77 | 0.76 | 154 | |

Figure 4.4

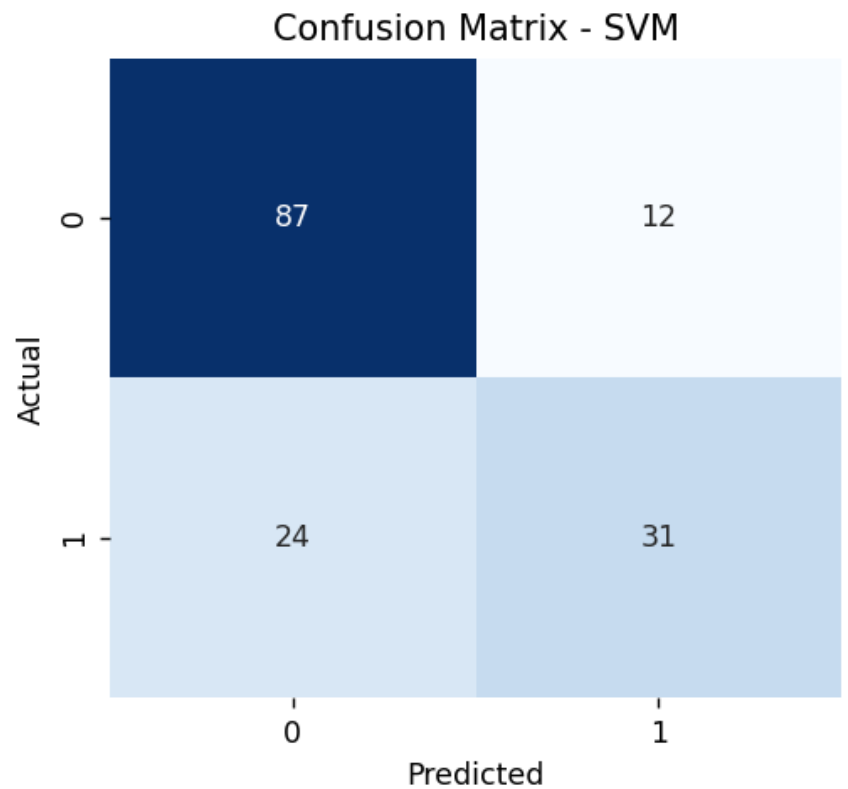


Figure 4.5

Based on the figure 4.4, an accuracy score of 76.66% for the Support Vector Machine (SVM) model indicates that approximately 76 out of every 100 predictions align with the true outcomes in the test dataset. SVMs excel in finding the optimal boundary that separates classes in complex datasets. This performance suggests that the SVM model has learned and generalized well to make correct predictions. However, similar to the Random Forest model, there's potential for further refinement. Fine-tuning SVM parameters like the choice of kernel or regularization strength could potentially enhance its predictive accuracy. Additionally, delving deeper into feature engineering or considering domain-specific insights could refine the model's ability to capture crucial patterns related to diabetes prediction. Evaluating additional performance metrics beyond accuracy, such as precision and recall, could offer a more nuanced understanding of the model's effectiveness, especially if the dataset exhibits class imbalances. This higher accuracy indicates promising performance, yet further optimization can refine the SVM's predictive power for diabetes diagnosis.

C) KNN:

| KNN Accuracy: 0.6623376623376623 | | | | | |
|----------------------------------|-----------|--------|----------|---------|--|
| | precision | recall | f1-score | support | |
| 0 | 0.75 | 0.71 | 0.73 | 99 | |
| 1 | 0.52 | 0.58 | 0.55 | 55 | |
| accuracy | | | 0.66 | 154 | |
| macro avg | 0.64 | 0.64 | 0.64 | 154 | |
| weighted avg | 0.67 | 0.66 | 0.67 | 154 | |

Figure 4.6

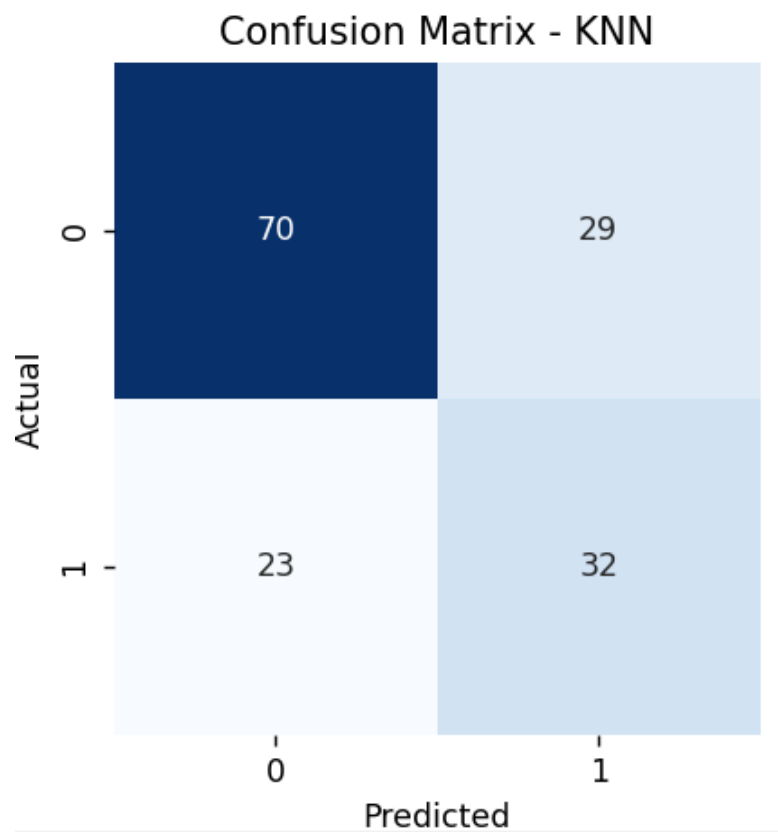
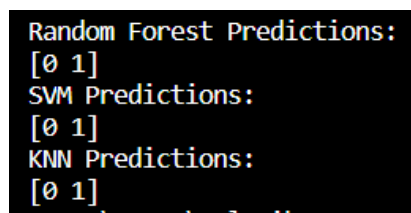


Figure 4.7

Based on the figure 4.6, An accuracy score of 66.23% for the K-Nearest Neighbors (KNN) model denotes that approximately 66 out of every 100 predictions made by the model align with the actual outcomes in the test dataset. KNN operates by classifying instances based on their similarity to neighboring data points, making it sensitive to local patterns within the dataset. While achieving a moderate accuracy, there's an opportunity for improvement. Optimizing the number of neighbors considered or exploring different distance metrics might enhance the model's predictive capability. Furthermore, conducting thorough feature

selection or engineering to extract more relevant information from the dataset could refine the KNN model's performance in diabetes prediction. Considering the specific context of diabetes diagnosis and understanding the clinical implications of correct predictions could guide further model enhancements. Exploring alternative evaluation metrics, especially in scenarios with imbalanced class distributions, might provide deeper insights into the model's efficacy. Although the accuracy is moderate, fine-tuning parameters and refining feature representation could potentially elevate the KNN model's performance in predicting diabetes outcomes.

4.2 Prediction and Decision Making



```
Random Forest Predictions:
[0 1]
SVM Predictions:
[0 1]
KNN Predictions:
[0 1]
```

Figure 4.8

In prediction and decision-making scenarios where the models output binary classes (0 or 1), such as in diabetes diagnosis, the three models—Random Forest, Support Vector Machine (SVM), and K-Nearest Neighbors (KNN).

A) Random Forest:

Description: Random Forest, through its ensemble of decision trees, offers diverse perspectives on predicting diabetes outcomes. It aggregates multiple tree-based predictions to make a final decision, providing robustness and an understanding of various feature influences on the prediction.

Summarization: It's a versatile model providing reasonably accurate predictions. It considers interactions among features and offers insights into feature importance, aiding in understanding the factors contributing to diabetes outcomes.

B) Support Vector Machine (SVM):

Description: SVM, known for creating optimal decision boundaries, excels in classifying diabetes outcomes by identifying the most effective separation between classes in a

high-dimensional space. It aims to maximize the margin between classes, offering robustness against overfitting.

Summarization: SVM delivers high accuracy and effective class separation. It's suitable for scenarios where precise delineation between diabetic and non-diabetic cases is essential, providing a clear decision boundary for diagnosis.

C) K-Nearest Neighbors (KNN):

Description: KNN relies on the similarity of instances to make predictions, classifying based on the majority vote of its neighboring points. It captures local patterns and makes predictions by considering the vicinity of data points.

Summarization: KNN, while demonstrating moderate accuracy, offers insights into local patterns within the dataset. It's valuable for recognizing localized trends or nuances in diabetes cases, although it may be sensitive to outliers or noise.

Decision-Making Insights:

- Ensemble Perspectives: Random Forest offers a collective view from multiple decision trees, providing a well-rounded prediction by considering diverse feature interactions.
- Optimal Separation: SVM draws clear boundaries between diabetic and non-diabetic cases, ideal for scenarios prioritizing precision in classification.
- Localized Patterns: KNN identifies localized trends within the dataset, offering insights into specific subsets or localized characteristics of diabetes cases.

Final Decision Strategy:

- Employ Random Forest for holistic insights into feature importance and interactions.
- Use SVM for precise class separation in scenarios requiring definitive diagnostic boundaries.
- Leverage KNN for recognizing localized patterns or specific subsets within the dataset, aiding nuanced understanding of diabetes variations.

4.4 Summary Chapter 4:

1. Random Forest:

- Accuracy: 72.07%
- Strengths: Demonstrates a decent predictive ability with its ensemble nature, managing overfitting, and handling various data types.
- Potential Improvement: Further parameter tuning and feature engineering could potentially enhance its accuracy and specificity for diabetes prediction.

2. Support Vector Machine (SVM):

- Accuracy: 76.66%
- Strengths: Shows promising predictive capabilities by optimizing class separation in complex datasets.
- Potential Improvements: Fine-tuning parameters and incorporating domain-specific insights might further improve its accuracy and applicability in diabetes diagnosis.

3. K-Nearest Neighbors (KNN)

- Accuracy: 66.23%
- Strength: Decent performance by classifying based on local similarities, capturing localized patterns within the data.
- Potential Improvements: Optimization of neighbor count, distance metrics, and feature representation could elevate its accuracy and robustness in diabetes prediction.

Overall Conclusion:

The SVM model exhibits the highest accuracy among the three models, showcasing promising predictive capabilities in diabetes outcome prediction. However, the Random Forest and KNN models also offer valuable insights despite their slightly lower accuracies. Each model has its strengths and areas for improvement, indicating the potential for further refinement to enhance their efficacy in predicting diabetes outcomes. Fine-tuning parameters, incorporating domain expertise, and exploring feature engineering avenues could collectively elevate the predictive performance of these models for practical application in diabetes diagnosis. Evaluating beyond accuracy metrics and considering domain-specific implications will be crucial for selecting the most suitable model in clinical settings.