Programming for Bioinformatics (SECB3203)

Title: Diabetic prediction using the machine learning

Final Report

Lecturer: DR.Nies Hui Wen

Client : DR Noor Hidayah Binti Zakaria

Group 18

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

TABLE OF CONTENTS

CHAPTER 1: INTRODUCTION

1.1 Overview

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.

1.2 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 healthcare and medical research. Diabetes is a prevalent, chronic ailment that affects millions of individuals across the globe. Timely and accurate detection, and effective management, are 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, integrating 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.

1.3 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:

The application of classification algorithms, including Support Vector Machine (SVM), Random Forest, and K-Nearest Neighbors (K-NN), in machine learning has the potential to improve the accuracy and cost-effectiveness of predicting diabetic disease based on patient data. This, in turn, could lead to enhanced patient outcomes and more efficient allocation of healthcare resources.

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.

1.4 Aim

This project aims to design and develop a classification of diabetes prediction using three different algorithms which are Support Vector Machine(SVM), Random Forest, and K-Nearest Neighbors (K-NN).

1.5 Objectives

The objectives 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 algorithms Support Vector Machine(SVM), Random Forest, and K-Nearest Neighbors (K-NN) in order to determine which dataset is in the normal or abnormal presence of the diabetes disease.
3. To evaluate the performance of the Support Vector Machine(SVM), Random Forest, and K-Nearest Neighbors (K-NN) in terms of accuracy and precision.

1.6 Scopes

The scopes 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.
4. Utilizing tools such as VS Code and Github.

1.7 Summary of Chapter 1

The summary of Chapter 1 is that 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 algorithms for classification methods in identifying normal and abnormal cases in the dataset based on the presence of diabetes. The algorithms we will use for this project are Support Vector Machine(SVM), Random Forest, and K-Nearest Neighbors (K-NN).

CHAPTER 2: RESEARCH METHODOLOGY

2.1 Flowchart of the proposed approach

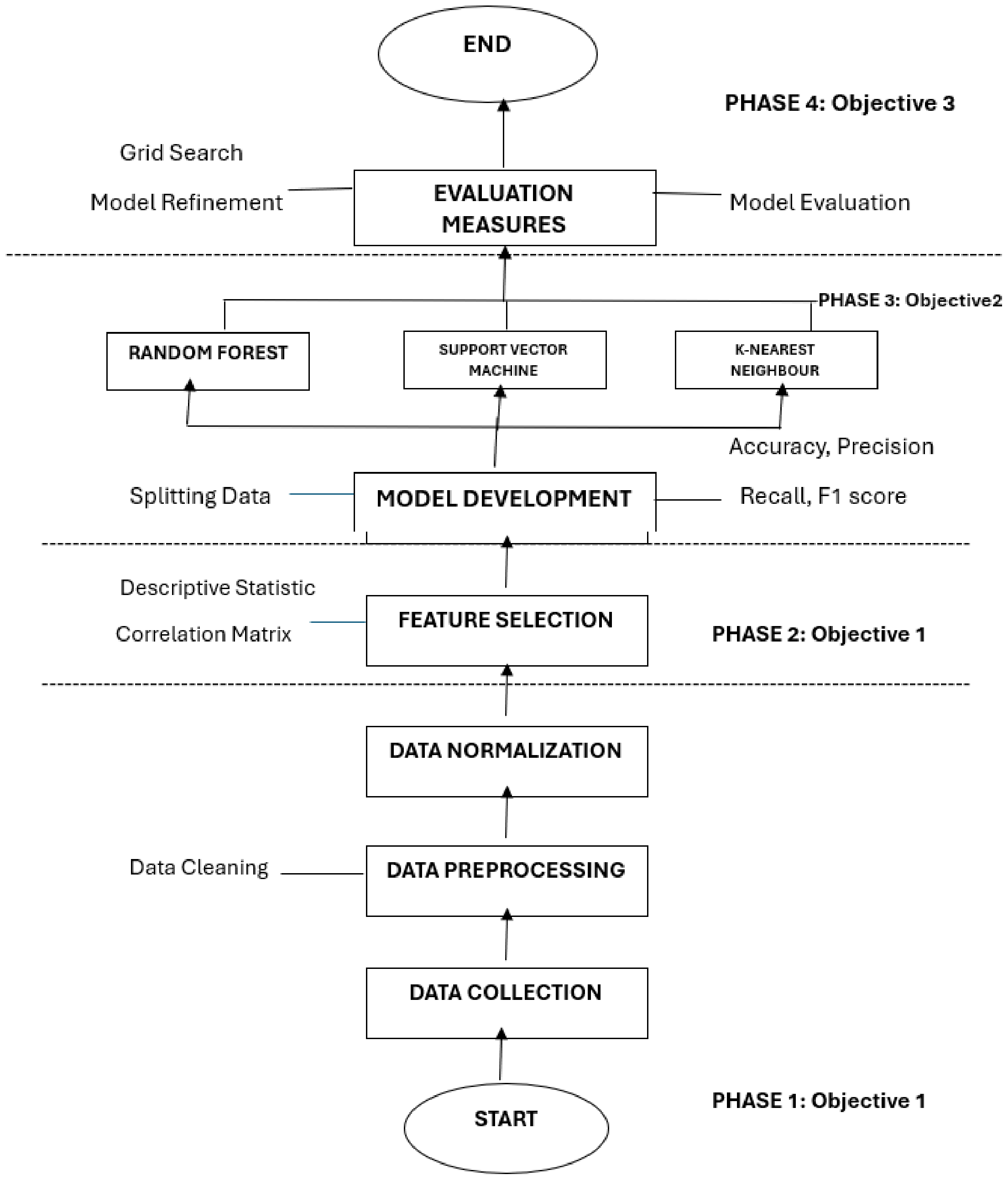


Figure 2.0

Based on Figure 2.0,

Phase 1: focusing on Objective 1, which involves studying the dataset, classification methods, and the diabetes domain to identify the domain, methods, and dataset used in Support Vector Machine (SVM), Random Forest, and K-Nearest Neighbors (K-NN). The process begins with the collection of data from the UCI Pima Indians Diabetes dataset. Subsequently, data preprocessing, or data cleaning, is carried out to identify any missing values in each column.

Following the data cleaning step, the next stage involves data normalization, a process aimed at standardizing the range of values in the dataset. This standardization is crucial for making the data comparable and interpretable across different features or variables. The entire Phase 1 is geared towards a comprehensive understanding of the dataset, classification methods, and the diabetes domain, laying the foundation for the subsequent study phases.

Phase 2: Objective 1 is centered on studying the dataset to select the most suitable features. This involves the implementation of feature selection through descriptive statistics and a correlation matrix. Descriptive statistics are utilized to summarize and describe the key features of the dataset. This statistical approach helps organize and simplify extensive data, enhancing its understandability and manageability.

Simultaneously, a correlation matrix is employed to identify and eliminate highly correlated features within the dataset. The objective is to mitigate redundancy and multicollinearity among features, positively influencing the performance of specific machine learning models. By systematically applying descriptive statistics and correlation matrix analysis, Phase 2 aims to refine the dataset, ensuring that only relevant and non-redundant features are retained for further analysis and model development.

Phase 3: Objective 2 is centered around the analysis and prediction of the diabetes dataset using three distinct algorithms: Support Vector Machine (SVM), Random Forest, and K-Nearest Neighbors (K-NN). The primary aim is to determine the presence of diabetes disease in the dataset, categorizing it as either normal or abnormal. The focus in this phase is on model development, achieved through the data splitting into training and testing sets.

During the model evaluation process, key metrics such as accuracy, precision, recall, and F1 score will be calculated. These metrics serve as benchmarks to compare the performance of the three algorithms which are SVM, Random Forest, and K-NN for prediction purposes. The analysis aims to identify the algorithm that provides the most accurate and reliable predictions for the presence of diabetes disease in the dataset.

Phase 4: Objective 3 is centered on the evaluation of the performance of three algorithms: Support Vector Machine (SVM), Random Forest, and K-Nearest Neighbors (K-NN). The evaluation focuses specifically on accuracy and precision, two critical metrics in assessing the effectiveness of classification models.

The evaluation process involves employing various evaluation measures and model evaluation techniques. This includes Grid Search, which is utilized to optimize hyperparameters for each algorithm, enhancing their predictive capabilities. Additionally, considerations are made for overfitting and under-fitting during the model evaluation stage.

Furthermore, model selection is undertaken as part of the refinement process. This involves choosing the most appropriate algorithm based on its performance in terms of accuracy and precision. The overall objective is to fine-tune and optimize the models through a comprehensive evaluation framework, ensuring their effectiveness in predicting the presence or absence of diabetes disease in the dataset.

2.2 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

CHAPTER 3 : DATA PREPROCESSING

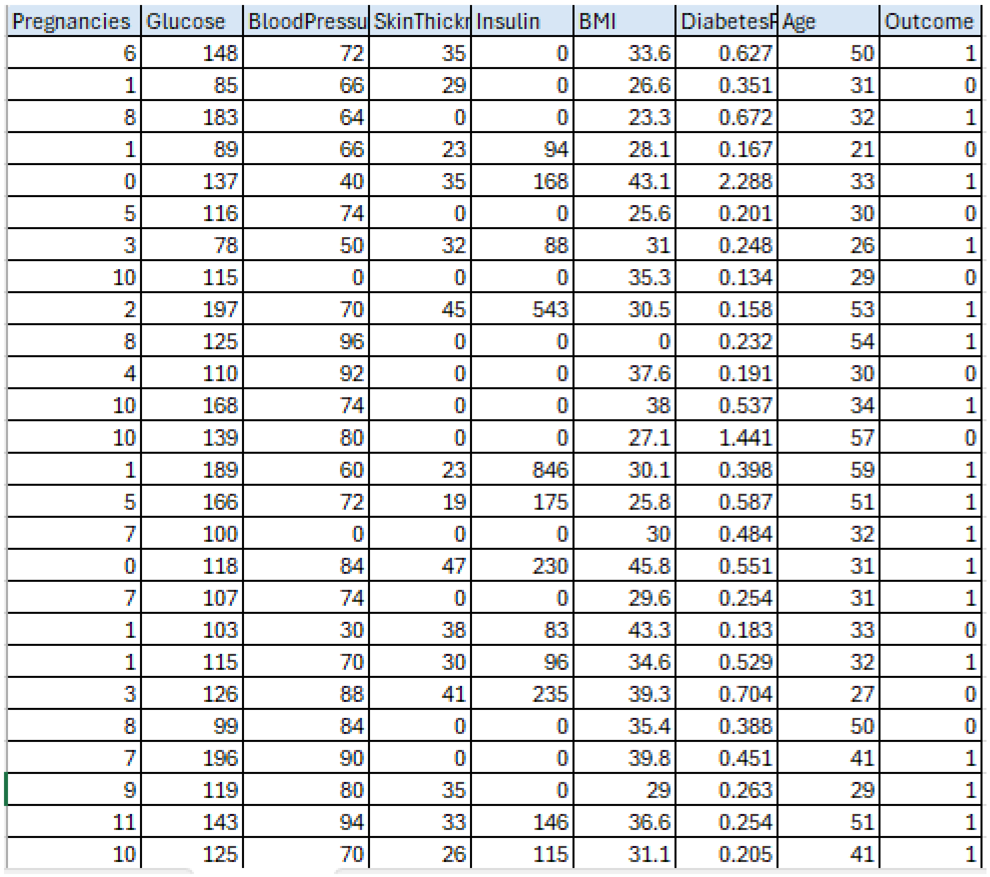
3.1 Importing Dataset

3.1.1 Understanding the dataset

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

Table 3.0



Pregnancies	Glucose	BloodPressu	SkinThickr	Insulin	BMI	DiabetesP	Age	Outcome
6	148	72	35	0	33.6	0.627	50	1
1	85	66	29	0	26.6	0.351	31	0
8	183	64	0	0	23.3	0.672	32	1
1	89	66	23	94	28.1	0.167	21	0
0	137	40	35	168	43.1	2.288	33	1
5	116	74	0	0	25.6	0.201	30	0
3	78	50	32	88	31	0.248	26	1
10	115	0	0	0	35.3	0.134	29	0
2	197	70	45	543	30.5	0.158	53	1
8	125	96	0	0	0	0.232	54	1
4	110	92	0	0	37.6	0.191	30	0
10	168	74	0	0	38	0.537	34	1
10	139	80	0	0	27.1	1.441	57	0
1	189	60	23	846	30.1	0.398	59	1
5	166	72	19	175	25.8	0.587	51	1
7	100	0	0	0	30	0.484	32	1
0	118	84	47	230	45.8	0.551	31	1
7	107	74	0	0	29.6	0.254	31	1
1	103	30	38	83	43.3	0.183	33	0
1	115	70	30	96	34.6	0.529	32	1
3	126	88	41	235	39.3	0.704	27	0
8	99	84	0	0	35.4	0.388	50	0
7	196	90	0	0	39.8	0.451	41	1
9	119	80	35	0	29	0.263	29	1
11	143	94	33	146	36.6	0.254	51	1
10	125	70	26	115	31.1	0.205	41	1

Table 3.0 illustrates the dataset obtained from UCI Pima Indians Diabetes, consisting of a total of 768 pregnant women (rows) and 8 features (columns).

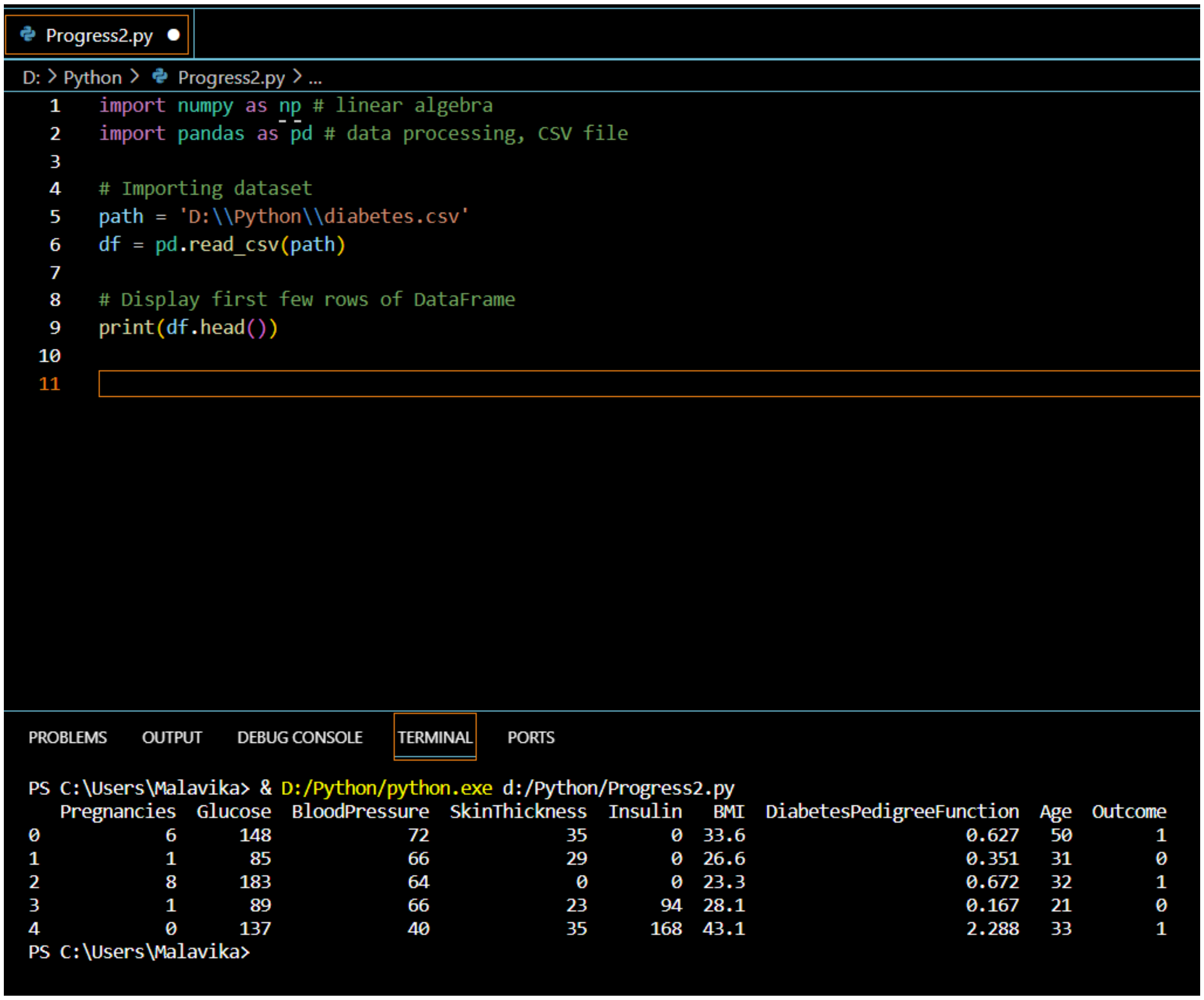
Table 3.1

Features	**Explanation**
Number of Time Pregnant	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.

Table 3.1 displays the features along with their explanations.

3.1.2 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.



Progress2.py

D: > Python > Progress2.py > ...

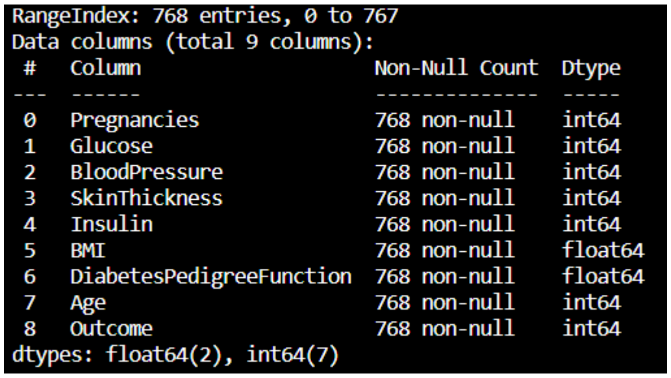
```python
import numpy as np # linear algebra
import pandas as pd # data processing, CSV file

# Importing dataset
path = 'D:\\Python\\diabetes.csv'
df = pd.read_csv(path)

# Display first few rows of DataFrame
print(df.head())

```

```
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

3.1.3 Getting Started Analyzing Data in Python

The data analyzation starts with getting know the number of rows and columns the dataset has. Rows and columns of the dataset



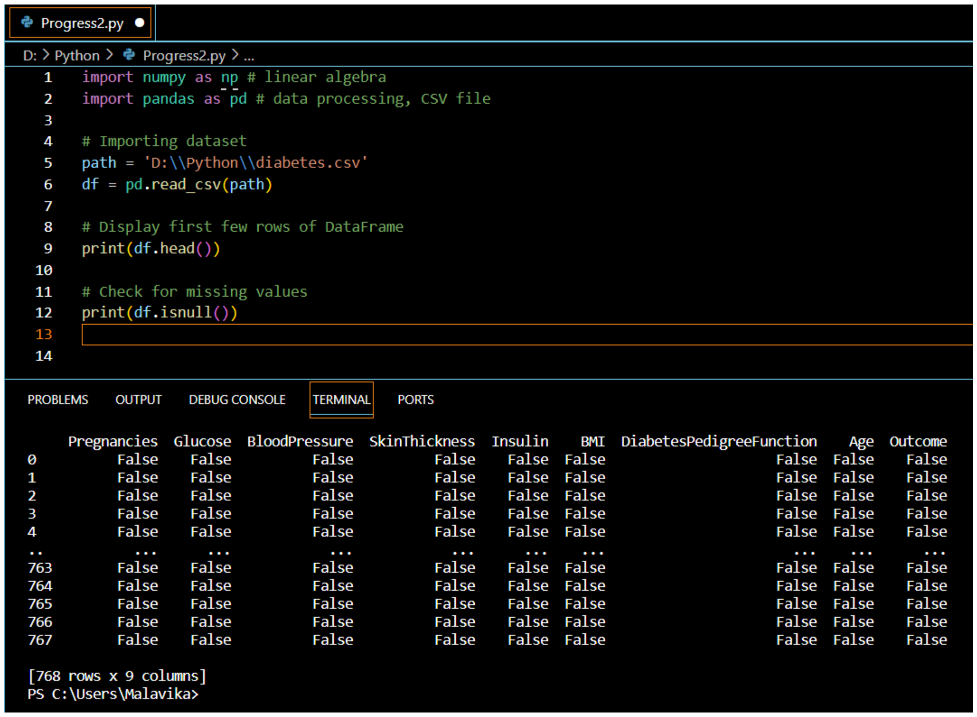
```
RangeIndex: 768 entries, 0 to 767
Data columns (total 9 columns):
 #   Column                    Non-Null Count  Dtype
---  ------                    --------------  -----
 0   Pregnancies               768 non-null    int64
 1   Glucose                   768 non-null    int64
 2   BloodPressure             768 non-null    int64
 3   SkinThickness             768 non-null    int64
 4   Insulin                   768 non-null    int64
 5   BMI                       768 non-null    float64
 6   DiabetesPedigreeFunction  768 non-null    float64
 7   Age                       768 non-null    int64
 8   Outcome                   768 non-null    int64
dtypes: float64(2), int64(7)
```

Figure 3.1 : Displaying total number of column and rows

3.2 Identifying and Handling Missing Values

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 > ...

```python
import numpy as np # linear algebra
import pandas as pd # data processing, CSV file

# Importing dataset
path = 'D:\\Python\\diabetes.csv'
df = pd.read_csv(path)

# Display first few rows of DataFrame
print(df.head())

# Check for missing values
print(df.isnull())

```

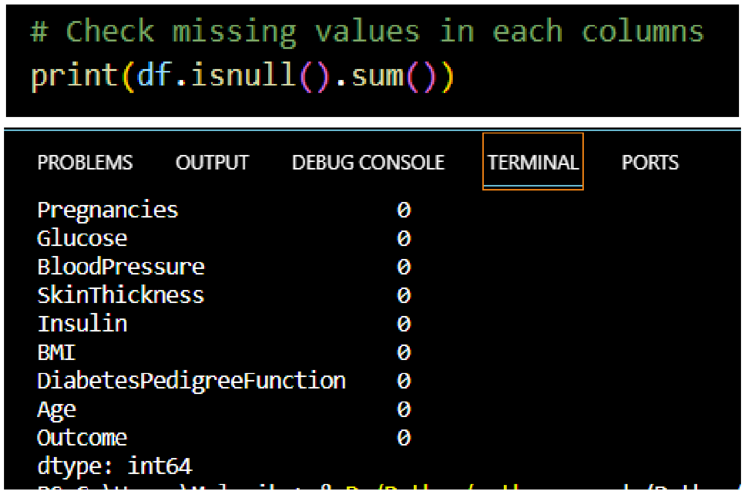
PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL PORTS

```
     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 3.2:Displaying there's no any missing values

3.3 Check missing values in each column

We checked each column separately in further detail by using the syntax print(df.isnull().sum). The outcome, which displays a count of 0 in each column, supports our previous conclusion that the dataset contains no null values. This column-by-column analysis is essential because it enables us to identify any particular features or attributes that might be missing values.



```python
# Check missing values in each columns
print(df.isnull().sum())
```

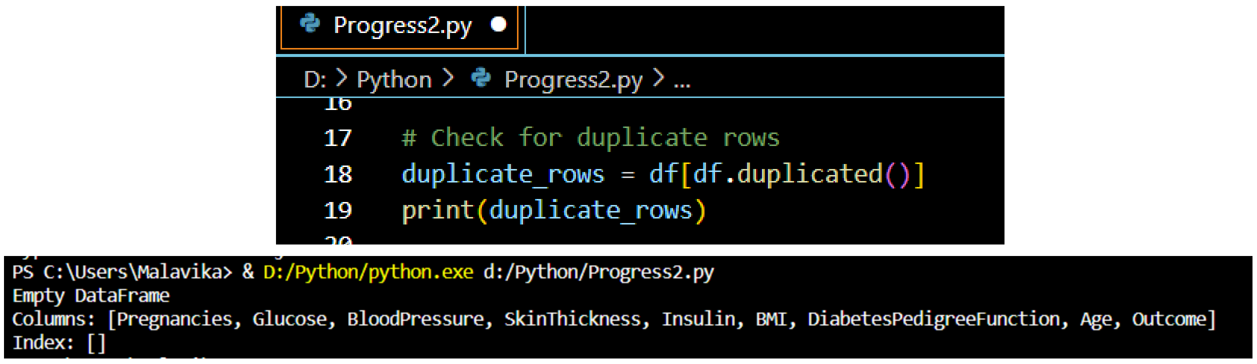
PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL PORTS

```
Pregnancies                 0
Glucose                     0
BloodPressure               0
SkinThickness               0
Insulin                     0
BMI                         0
DiabetesPedigreeFunction    0
Age                         0
Outcome                     0
dtype: int64
```

Figure 3.3:No missing values in each column

3.4 Check for duplicate rows

The next step in data preprocessing is to check for duplicate rows because duplicate entries can cause inaccurate outcomes. We checked for any duplicate rows in the dataset and we found that there are no duplicate rows.



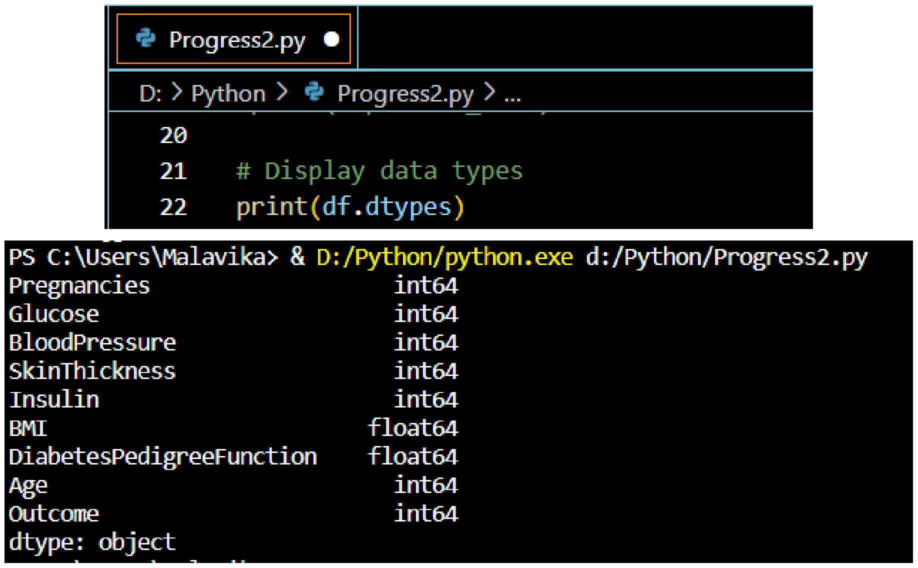
```python
# Check for duplicate rows
duplicate_rows = df[df.duplicated()]
print(duplicate_rows)
```

```
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 3.4 : No duplicate values

3.5 Data Formatting

The analysis then proceeded with getting to know the data types of each column. By using syntax print(df.dtypes) we can display data types for each column. Our dataset contains int64 representing integer data type and float64 representing floating point data type.



```python

# Display data types
print(df.dtypes)
```

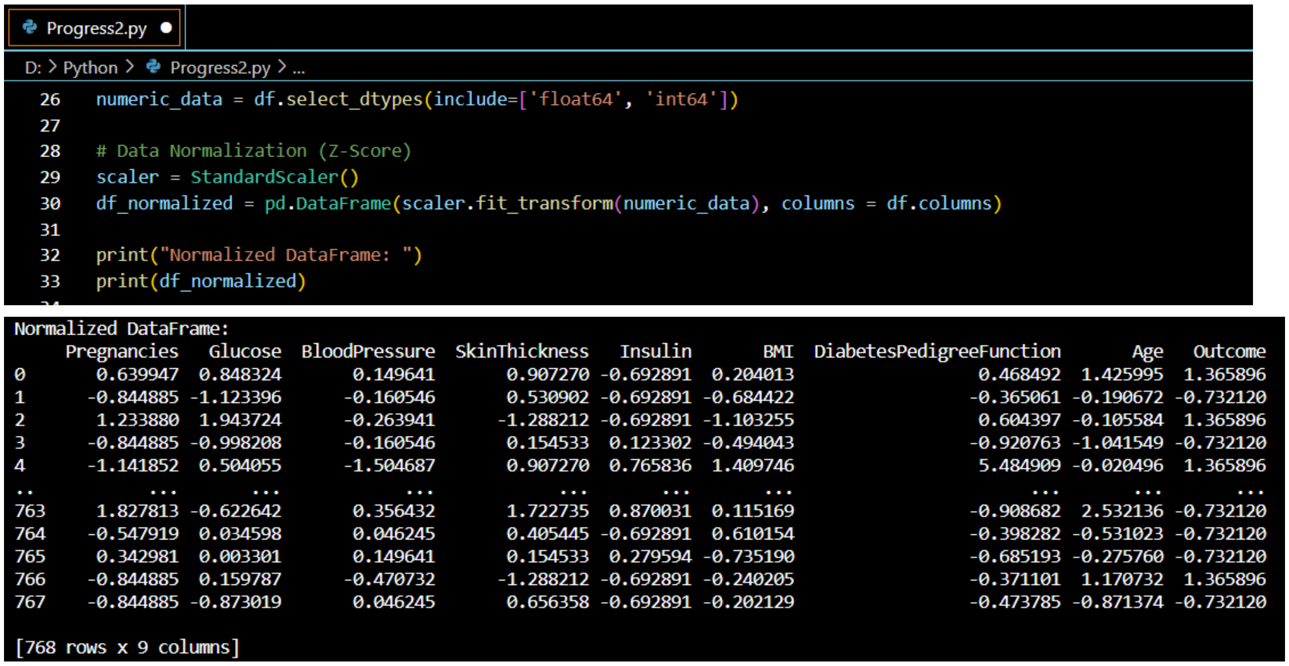
```
PS C:\Users\Malavika> & D:/Python/python.exe d:/Python/Progress2.py
Pregnancies                   int64
Glucose                       int64
BloodPressure                 int64
SkinThickness                 int64
Insulin                       int64
BMI                         float64
DiabetesPedigreeFunction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

Figure 3.5 Display data types in each column.

3.6 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.



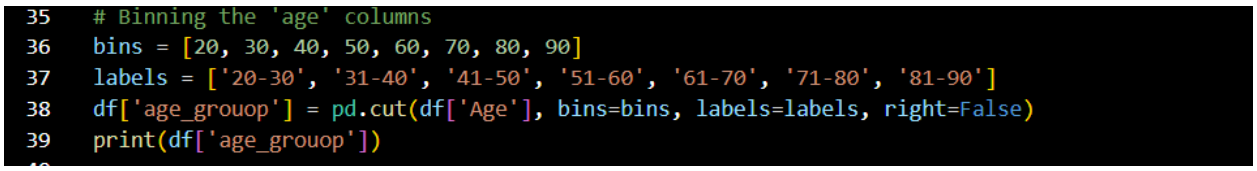
```python
numeric_data = df.select_dtypes(include=['float64', 'int64'])

# Data Normalization (Z-Score)
scaler = StandardScaler()
df_normalized = pd.DataFrame(scaler.fit_transform(numeric_data), columns = df.columns)

print("Normalized DataFrame: ")
print(df_normalized)
```

```
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]
```

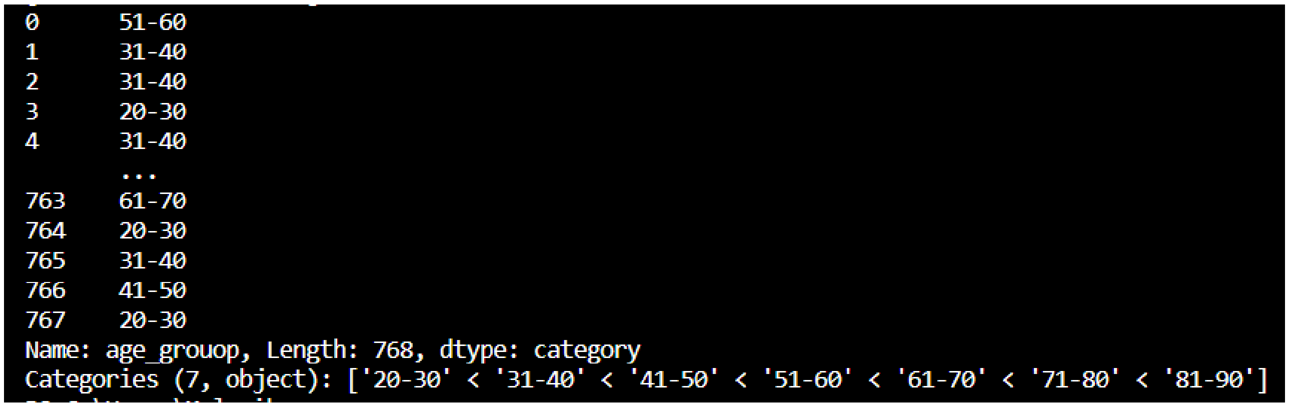
Figure 3.6 :Data Normalization: Centering output

3.7 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.



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



```
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']
```

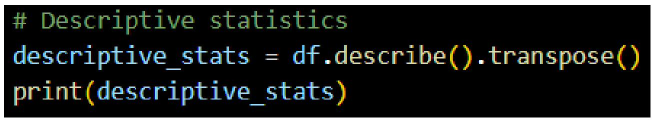
Figure 3.7 :Data binning (Age group)

3.8 Summary of Chapter 3

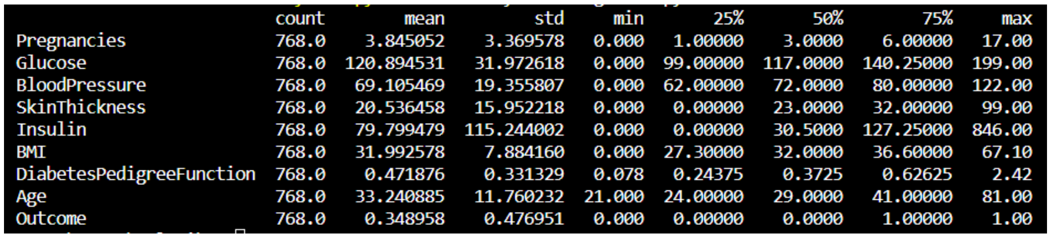
This chapter covers fundamental steps in Python for preparing and analyzing data effectively in data science workflows. We start by stressing the importance of clean, well-structured data for meaningful insights. Exploring multiple data import methods using Pandas, we ensure smooth handling of various 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: Exploratory Data Analysis

4.1 Descriptive Statistics



```
# Descriptive statistics
descriptive_stats = df.describe().transpose()
print(descriptive_stats)
```



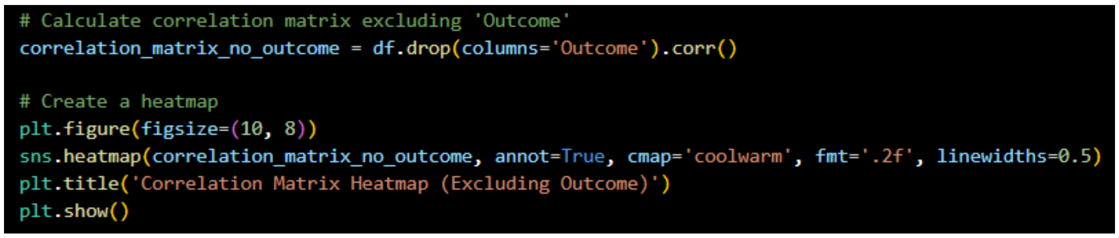
```
                          count        mean         std     min       25%       50%        75%     max
Pregnancies               768.0    3.845052    3.369578   0.000   1.00000    3.0000    6.00000   17.00
Glucose                   768.0  120.894531   31.972618   0.000  99.00000  117.0000  140.25000  199.00
BloodPressure             768.0   69.105469   19.355807   0.000  62.00000   72.0000   80.00000  122.00
SkinThickness             768.0   20.536458   15.952218   0.000   0.00000   23.0000   32.00000   99.00
Insulin                   768.0   79.799479  115.244002   0.000   0.00000   30.5000  127.25000  846.00
BMI                       768.0   31.992578    7.884160   0.000  27.30000   32.0000   36.60000   67.10
DiabetesPedigreeFunction  768.0    0.471876    0.331329   0.078   0.24375    0.3725    0.62625    2.42
Age                       768.0   33.240885   11.760232  21.000  24.00000   29.0000   41.00000   81.00
Outcome                   768.0    0.348958    0.476951   0.000   0.00000    0.0000    1.00000    1.00
```

Figure 4.1 shows descriptive statistics

Descriptive statistics help in summarizing the main characteristics of a dataset, such as central tendency (mean, median, mode), variability (range, variance, standard deviation), and distribution.

4.2 Correlation Matrix

A correlation matrix is a table that shows the correlation coefficients between many variables. Each cell in the table represents the correlation between two variables. Correlation, in the context of statistics, measures the strength and direction of a linear relationship between two variables.



```
# Calculate correlation matrix excluding 'Outcome'
correlation_matrix_no_outcome = df.drop(columns='Outcome').corr()

# Create a heatmap
plt.figure(figsize=(10, 8))
sns.heatmap(correlation_matrix_no_outcome, annot=True, cmap='coolwarm', fmt='.2f', linewidths=0.5)
plt.title('Correlation Matrix Heatmap (Excluding Outcome)')
plt.show()
```

Figure 4.2 shows the code for correlation matrix for feature selection

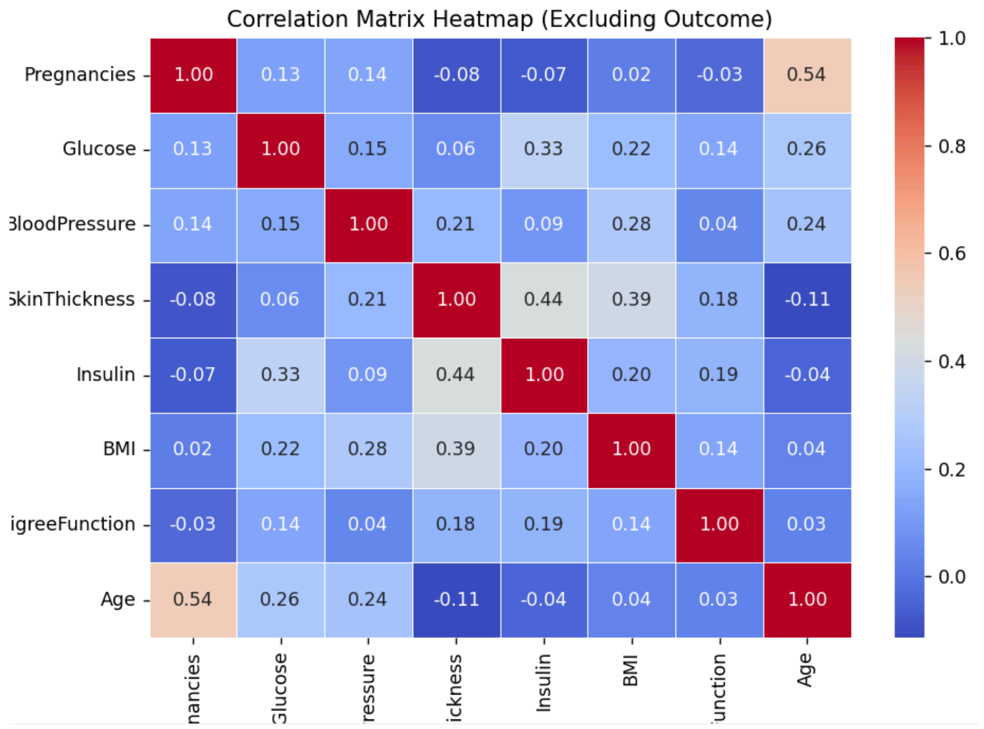


Figure 4.3 shows correlation matrix in heatmap

Based on the figure 4.3, Each entry in the correlation matrix is a correlation coefficient. The most commonly used correlation coefficient is the Pearson correlation coefficient, denoted by "r." It ranges from -1 to 1, where -1 indicates a perfect negative linear relationship, 1 indicates a perfect positive linear relationship, and 0 indicates no linear relationship.The correlation matrix is symmetric, meaning that the correlation between variable A and variable B is the same as the correlation between variable B and variable A.The diagonal elements of the correlation matrix always have a correlation coefficient of 1 because a variable has a perfect correlation with itself. Positive values of the correlation coefficient indicate a positive linear relationship, while negative values indicate a negative linear relationship. A correlation coefficient close to 0 suggests a weak or no linear relationship.The absolute value of the correlation coefficient indicates the strength of the relationship. Values closer to 1 (either positive or negative) suggest a stronger linear relationship, while values closer to 0 suggest a weaker relationship.

Variables with values exceeding 0.60 are considered the strongest features. However, in this case, we did not identify any features surpassing 0.50. Consequently, we are utilizing features with values close to the threshold

Selected features from the heatmap:
- Pregnancies (0.54)
- SkinThickness (0.44)
- Insulin (0.44)

- BMI (0.39)
- Age (0.54)

4.3 Summary of Chapter 4

In this analysis, descriptive statistics were employed to succinctly summarize key characteristics of the dataset, encompassing measures of central tendency, variability, and distribution. Moving into the examination of the correlation matrix, the Pearson correlation coefficient, denoted as "r," was utilized to gauge the strength and direction of linear relationships between variables. The matrix, exhibiting symmetry, highlighted the consistency of correlations between pairs of variables. Notably, the diagonal elements maintained a correlation coefficient of 1, indicative of a perfect correlation between each variable and itself. Positive coefficients signaled positive linear relationships, negative coefficients indicated negative relationships, while values close to 0 suggested weaker or nonexistent linear connections.

Shifting focus to feature selection, a customary threshold of 0.60 for strong features was applied. However, in this instance, no features surpassed the 0.50 threshold. Consequently, features with values proximal to the threshold were chosen for further consideration. The selected features, as gleaned from the heatmap, include Pregnancies (0.54), SkinThickness (0.44), Insulin (0.44), BMI (0.39), and Age (0.54). These features, though not meeting the typical 0.60 threshold, exhibit correlations close to 0.50, offering valuable insights for subsequent analyses or modeling within the context of the UCI Pima Indians Diabetes dataset.

Chapter 5: Model Development

5.1 Machine Learning Model Development Flowchart

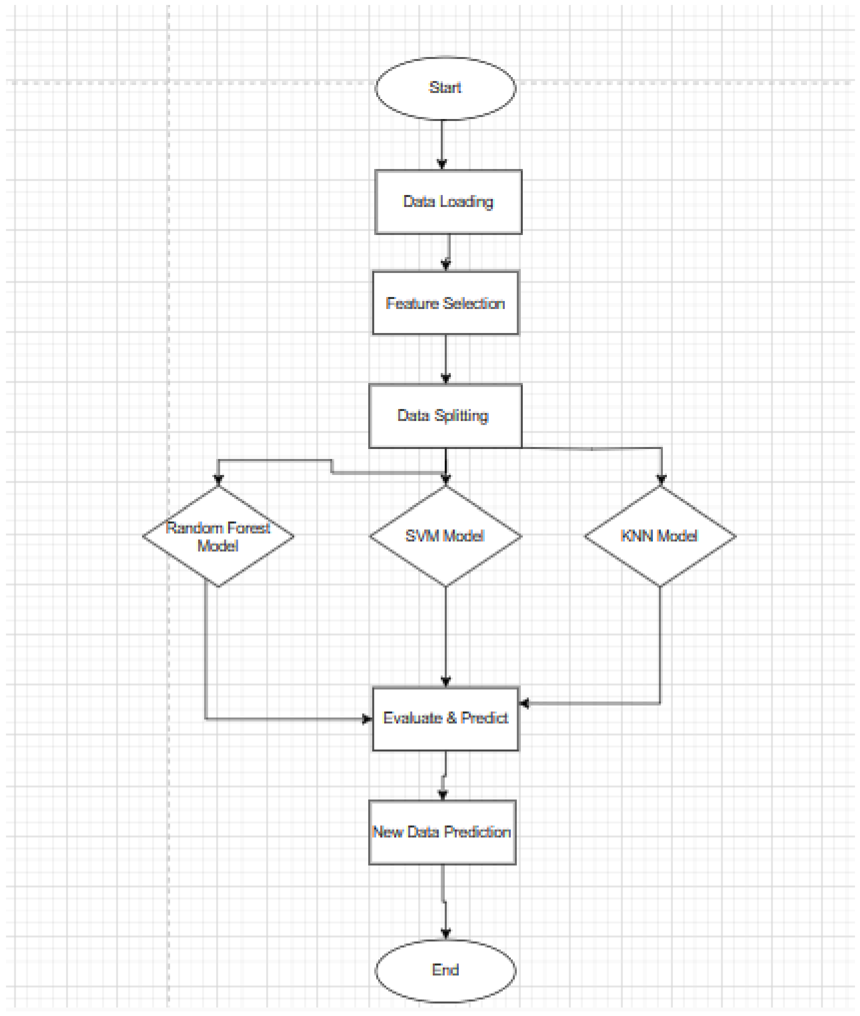
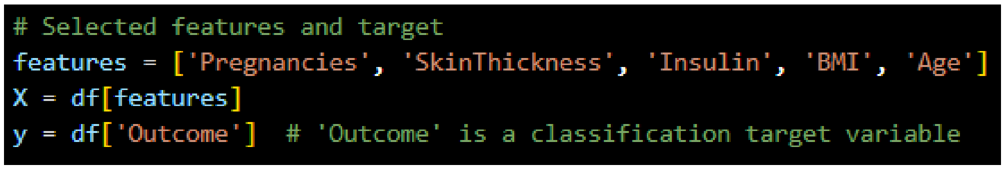


Figure 5.0

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.

5.2 Model Training

Selected Features and Target

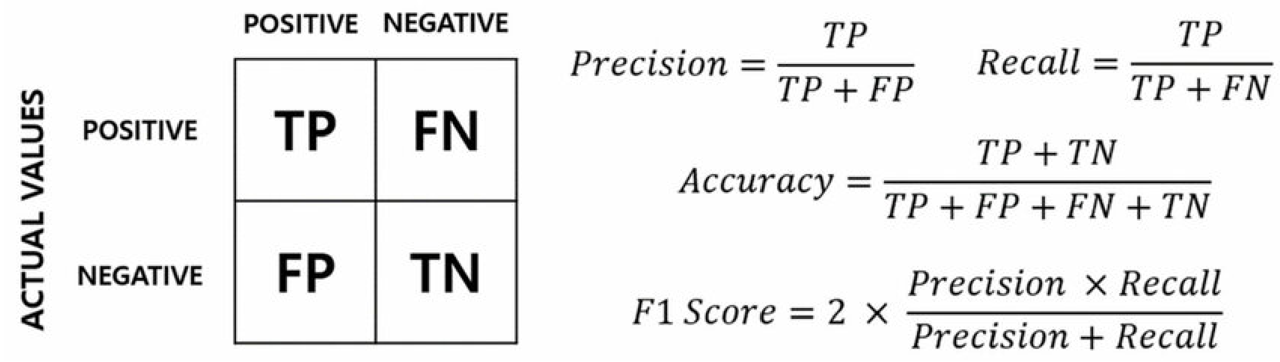


```
# Selected features and target
features = ['Pregnancies', 'SkinThickness', 'Insulin', 'BMI', 'Age']
X = df[features]
y = df['Outcome']  # 'Outcome' is a classification target variable
```

Figure 5.1

Figure 5.1 shows the selected features to measure performance of SVM, random forest and K-NN.

5.2.1 Performance Metrics



$$Precision = \frac{TP}{TP + FP} \qquad Recall = \frac{TP}{TP + FN}$$

$$Accuracy = \frac{TP + TN}{TP + FP + FN + TN}$$

$$F1\,Score = 2 \times \frac{Precision \times Recall}{Precision + Recall}$$

Figure 5.2

Based on the figure 5.2,

The **Confusion Matrix r**esembles :

- True Positive(TP) signifies how many positive class samples your model predicted correctly.
- True Negative(TN) signifies how many negative class samples your model predicted correctly.
- False Positive(FP) signifies how many negative class samples your model predicted incorrectly. This factor represents Type-I error in statistical nomenclature. This error positioning in the confusion matrix depends on the choice of the null hypothesis.

- False Negative(FN) signifies how many positive class samples your model predicted incorrectly. This factor represents Type-II error in statistical nomenclature. This error positioning in the confusion matrix also depends on the choice of the null hypothesis.

Accuracy : Classification accuracy is perhaps the simplest metric to use and implement and is defined as the number of correct predictions divided by the total number of predictions

Precision : Precision is the ratio of true positives and total positives predicted. The precision metric focuses on Type-I errors(FP). A Type-I error occurs when we reject a true null Hypothesis(H^0)**.**

Recall :A Recall is essentially the ratio of true positives to all the positives in ground truth.The recall metric focuses on type-II errors(FN). A type-II error occurs when we accept a false null hypothesis(H^0)**.**

F1-score: The F1-score metric uses a combination of precision and recall. In fact, the F1 score is the harmonic mean of the two.

5.2.2 Random Forest:

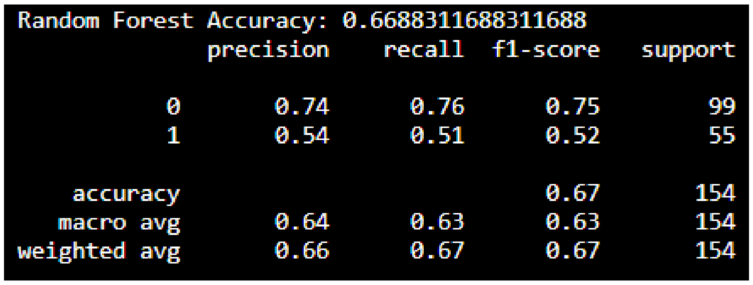
FORMULA FOR RANDOM FOREST:

$$Gini = 1 - \sum_{i=1}^{C} (p_i)^2$$

Figure 5.3

Pi: represents the probability of observing class
i at a particular node.
c is the number of classes in the dataset.

The Gini index, used in decision trees and Random Forests, measures the impurity or disorder of a node in a decision tree. The formula calculates the likelihood of a misclassification if a random sample were to be incorrectly labeled based on the distribution of classes in a node.



```
Random Forest Accuracy: 0.6688311688311688
              precision    recall  f1-score   support

           0       0.74      0.76      0.75        99
           1       0.54      0.51      0.52        55

    accuracy                           0.67       154
   macro avg       0.64      0.63      0.63       154
weighted avg       0.66      0.67      0.67       154
```

Figure 5.4

Based on the figure 5.4, an accuracy of 0.6689 for the Random Forest model suggests that nearly 67 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.

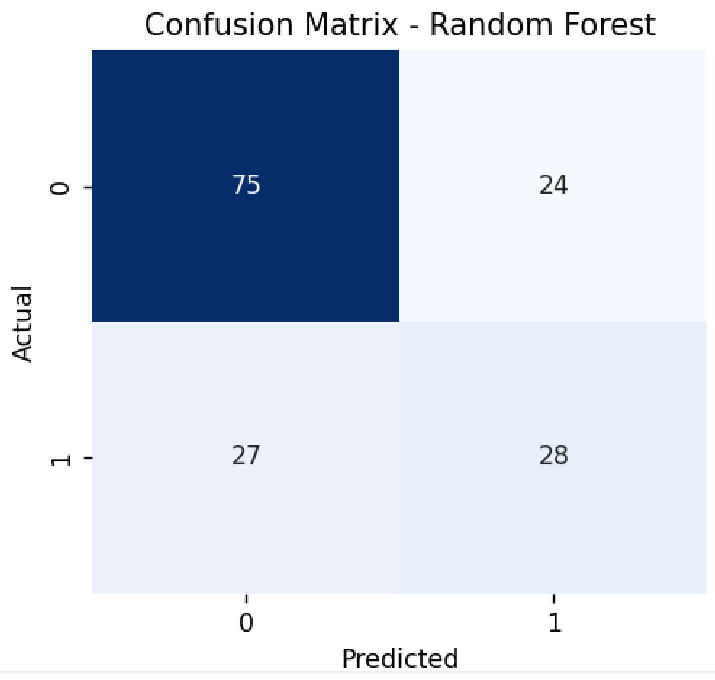


Figure 5.5

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.

5.2.3 Support Vector Machine (SVM):

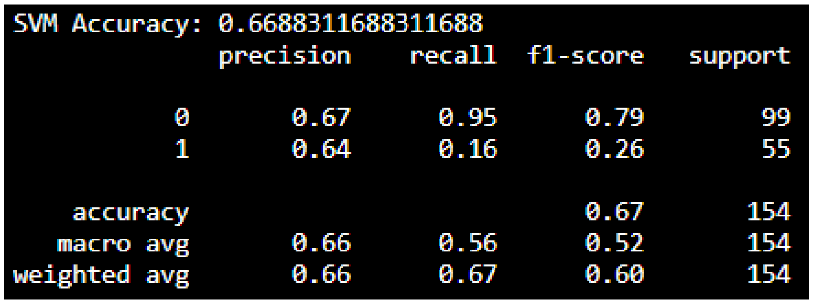
Formula :

$$f(x) = \operatorname{sign}\left(\left(\sum_{j=1}^{n} w_j \cdot x_{ij}\right) + b\right)$$

Figure 5.6

Here:

- $f(x)$ is the decision function that predicts the class of a new data point x.
- w_j are the weights assigned to each feature.
- b is the bias term.
- $\operatorname{sign}(\cdot)$ is the sign function that returns -1 for negative values, 0 for zero, and 1 for positive values.



```
SVM Accuracy: 0.6688311688311688
              precision    recall  f1-score   support

           0       0.67      0.95      0.79        99
           1       0.64      0.16      0.26        55

    accuracy                           0.67       154
   macro avg       0.66      0.56      0.52       154
weighted avg       0.66      0.67      0.60       154
```

Figure 5.7

Based on the figure 5.7, an accuracy score of 0.6689 for the Support Vector Machine (SVM) model indicates that approximately 67 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.

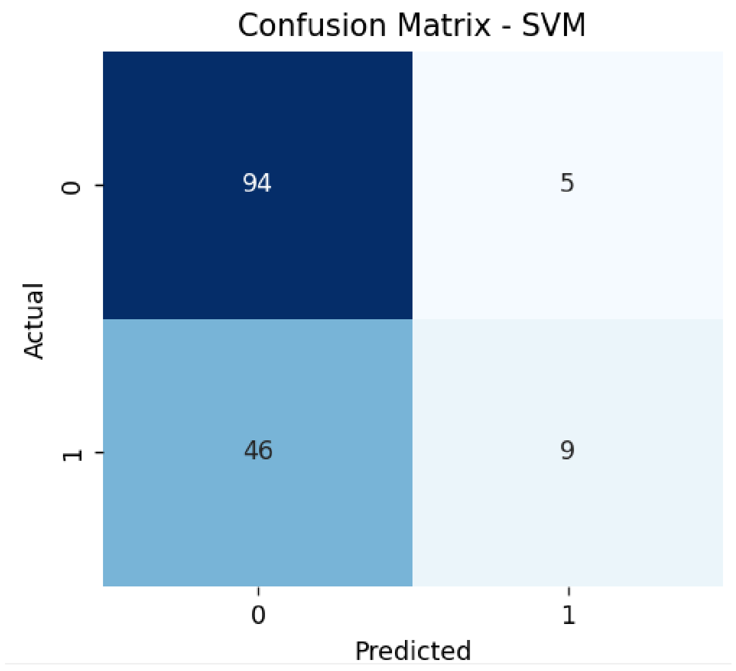


Figure 5.8

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.

5.2.4 K-Nearest Neighbor (KNN) :

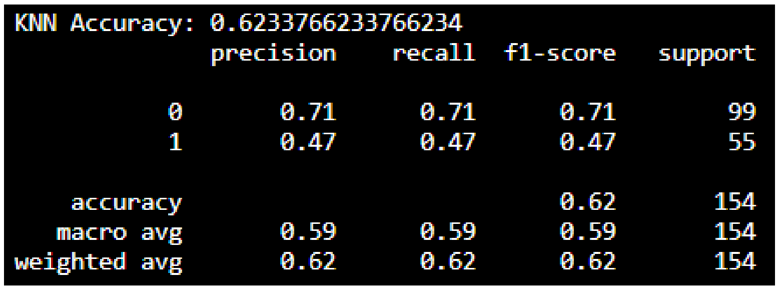
Formula :

$$y = \operatorname{argmax}_c \left(\sum_{i=1}^{k} I(y_i = c) \right)$$

Figure 5.9

Here :

- y is the predicted class label for the new data point.
- c represents each unique class in the dataset.
- $I(\cdot)$ is the indicator function that returns 1 if the condition is true and 0 otherwise.
- argmax_c finds the class c for which the sum inside the parentheses is maximized.



```
KNN Accuracy: 0.6233766233766234
              precision    recall  f1-score   support

           0       0.71      0.71      0.71        99
           1       0.47      0.47      0.47        55

    accuracy                           0.62       154
   macro avg       0.59      0.59      0.59       154
weighted avg       0.62      0.62      0.62       154
```

Figure 5.10

Based on the figure 5.10, an accuracy score of 0.6233 for the K-Nearest Neighbors (KNN) model denotes that approximately 62 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

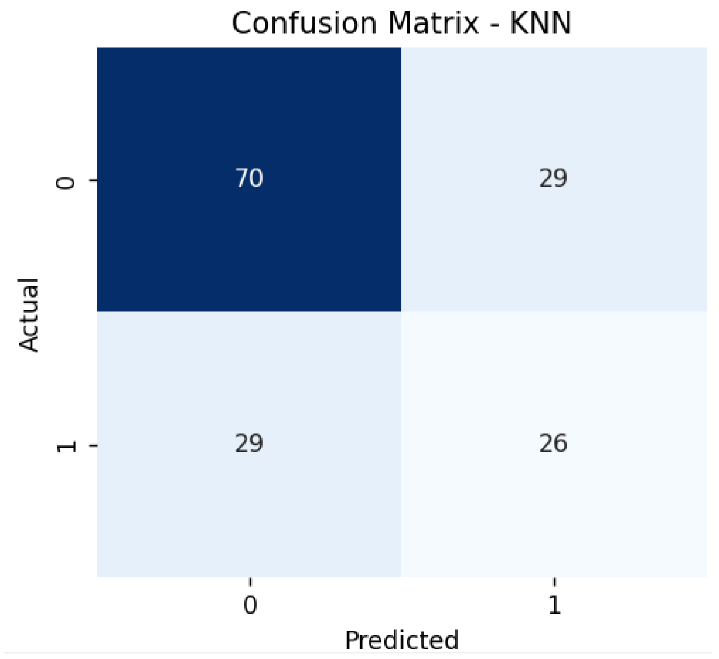
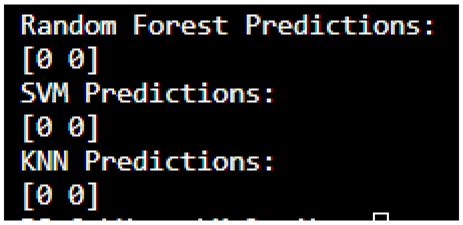


Figure 4.10

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.

5.3 Prediction and Decision Making



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

Figure 4.11

Based on the above figure, it typically means that the model is assigning the input to the class represented by the label "0" for both of its output classes. In a binary classification setting, where there are two classes [0, 0] as the prediction indicates that the model is predicting the instance to belong to the class labeled as "0" with high confidence.

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.

5.3.1 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.

5.3.2 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.

5.3.3 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.

5.4 Summary of Chapter 5

1. Random Forest:

- Accuracy: 66.88%
- Precision: 0.74
- 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: 66.88%
- Precision : 0.67
- 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: 62.33%
- Precision : 0.71
- 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 Random Forest model exhibits the highest accuracy among the three models, showcasing promising predictive capabilities in diabetes outcome prediction. However, the SVM 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.

CHAPTER 6: MODEL EVALUATION

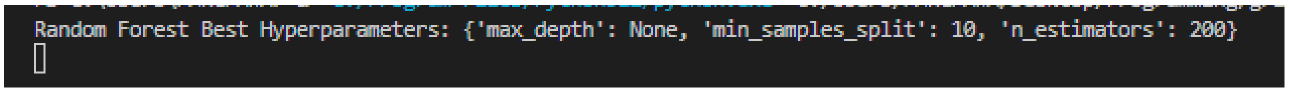
6.1 Model evaluation

Model evaluation is a critical aspect of machine learning and statistical modeling. It involves assessing the performance and effectiveness of a predictive model using various techniques and metrics. The primary goal is to understand how well a model generalizes to new, unseen data and whether it accomplishes its intended task accurately and reliably.

The process of model evaluation is iterative, involving the refinement of models, adjustment of hyperparameters, and comparison of different algorithms to identify the most effective solution for a specific problem domain. Ultimately, model evaluation aims to ensure that the selected model is robust, reliable, and capable of making accurate predictions when applied to new, unseen data.

6.2 Grid Search

It is a hyperparameter tuning technique, which is more about optimizing model performance by finding the best hyperparameters. It's not directly an evaluation technique but is often used as part of model evaluation to improve a model's performance.



```
Random Forest Best Hyperparameters: {'max_depth': None, 'min_samples_split': 10, 'n_estimators': 200}
```

Figure 6.0

Based on Figure 6.0, the Random Forest algorithm emerged as the optimal choice for predicting diabetes based on the provided dataset and features, as determined through a meticulous grid search of hyperparameters. Random Forest, being an ensemble method, constructs a multitude of decision trees and amalgamates their predictions. During the grid search, the algorithm's performance was assessed across various configurations of hyperparameters, namely, the number of trees in the forest (`n_estimators`), the maximum depth of each tree (`max_depth`), and the minimum number of samples required to split an internal node (`min_samples_split`).

The chosen configuration of hyperparameters reflects a careful balance between model complexity and generalization. The optimal number of trees, a critical factor in Random Forest's efficacy, was identified, ensuring that the algorithm captures intricate patterns without succumbing to overfitting. Additionally, the ideal depth of each tree was determined to strike the right balance between complexity and simplicity, crucial for preventing both underfitting and overfitting. The minimum samples required for node splits were carefully tuned to refine the granularity of decision-making in the model.

The superior performance of Random Forest can be attributed to its ability to handle complex relationships within the data, mitigate overfitting through ensemble learning, and fine-tune of hyperparameters. The specific characteristics of the dataset, coupled with the intricate interplay between features, likely influenced the selection of these hyperparameters, resulting in a predictive model that excels in forecasting diabetes outcomes.

6.3 Over-fitting, under-fitting, and model selection

A) K-Nearest Neighbor(KNN)

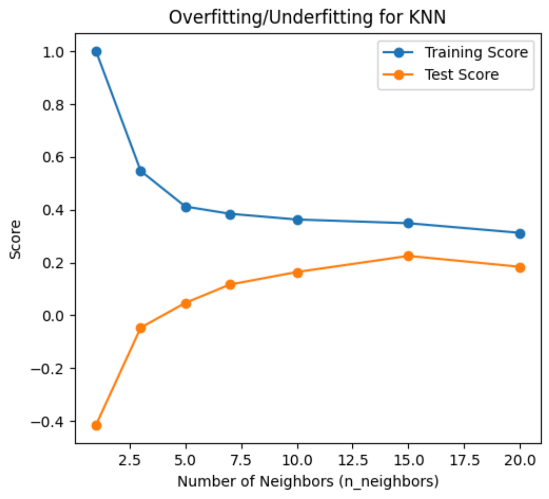


Figure 6.1

Based on figure 6.1, Training Scores (1.00 - 0.40): The wide range of training scores suggests varying performance levels on the training data. Scores declining from perfect (1.00) to moderate (0.40) imply that the model's ability to fit the training data fluctuates significantly.

Testing Scores (-0.4 to 0.2): The negative score suggests substantial overfitting or serious model misfitting during cross-validation. The range from negative values to low positive scores indicates poor performance, suggesting that the model doesn't generalize well to unseen data.

B) SUPPORT VECTOR MACHINE (SVM)

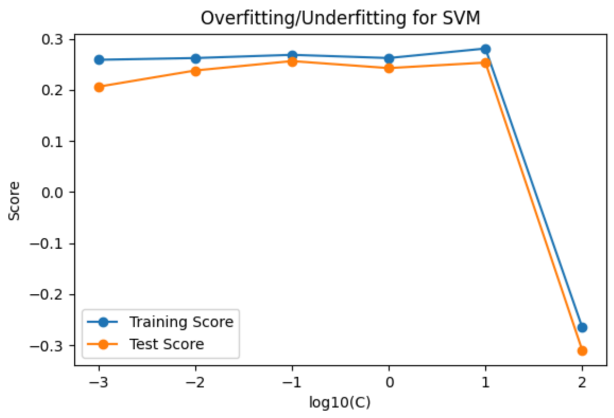


Figure 6.2

Based on figure 6.2, Training Scores (0.25 to -0.25): The scores ranging from 0.25 to -0.25 indicate varying and generally poor performance on the training data. Such low scores suggest the model's inability to capture patterns within the training set consistently.

Testing Scores (0.2 to 2): The cross-validation scores ranging from 0.2 to 2 display a wide range of performance on different subsets of the data during validation. The scores, although showing variability, encompass both moderately low and high values.

C) Random Forest

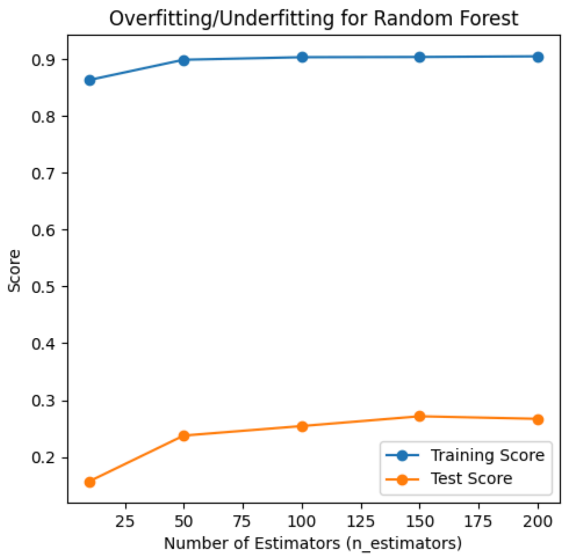


Figure 6.3

Based on figure 6.3, Training Scores (0.85 to 0.90): The consistent and relatively high training scores, ranging from 0.85 to 0.90, suggest that the model fits the training data well. These scores indicate a good ability to capture patterns and details present in the training set.

Testing Scores (0.1 to 0.3): The cross-validation scores, ranging from 0.1 to 0.3, indicate moderate performance on unseen data. While the scores are notably lower than the training scores, they still suggest a moderate level of generalization ability to new data.

6.4 Summary of Chapter 6

The KNN model appears to suffer from severe overfitting or a lack of model flexibility, as indicated by the negative cross-validation scores and a wide range between training and cross-validation scores. Addressing overfitting by potentially reducing the model's complexity, employing feature selection, or tuning hyperparameters might be necessary to improve its generalization ability. Additionally, considering alternative models or algorithms might also be beneficial to achieve better performance on unseen data.

The Random Forest model demonstrates a disparity between its performance on the training and cross-validation sets: Potential Overfitting: The considerable gap between the high training scores and the lower cross-validation scores suggests potential overfitting. The model might be overly complex and fail to generalize well to new data. Moderate Generalization: The moderate cross-validation scores, although lower than the training scores, suggest that the model can still generalize reasonably well but might not capture the same level of detail seen in the training data.

The SVM model's performance reveals inconsistent fitting to the training data, reflected by generally poor training scores. The variability observed in cross-validation scores might suggest the model's sensitivity to different subsets of the data. To improve the SVM model: Tackle Underfitting: Address the model's inadequate fitting to the training data by exploring more complex kernels or adjusting hyperparameters to increase model flexibility.Fine-tuning: Refine hyperparameters such as C or gamma to improve model performance and better capture patterns in the data.Feature Engineering: Revisit feature selection or engineering to ensure the model is learning from the most informative features available.

7.0 Conclusion

In conclusion, our investigation aimed to address the challenge of enhancing diabetes prediction through the application of machine learning algorithms—Support Vector Machine (SVM), Random Forest, and K-Nearest Neighbors (K-NN)—utilizing patient data, particularly among pregnant women. Diabetes, being a prevalent and chronic health condition, demands efficient and cost-effective predictive models for early detection and management.

Our findings indicate that among the tested algorithms, Random Forest demonstrated the highest accuracy, making it a robust choice for predicting diabetes based on variables such as 'Pregnancies,' 'SkinThickness,' 'Insulin,' 'BMI,' and 'Age.' This model holds promise for identifying at-risk individuals early on, facilitating timely interventions and personalized treatment plans.

While SVM also exhibited competitive accuracy, Random Forest outperformed it, and K-Nearest Neighbors lagged slightly behind. The predictive capabilities of these models have significant implications for improving patient outcomes and optimizing healthcare resource allocation.

This research aligns with the broader goal of leveraging machine learning techniques to enhance the efficiency of diabetes prediction, contributing to early intervention strategies, personalized healthcare, and potential cost savings within healthcare systems. By focusing on pregnant women, our models aim to address a specific demographic group, providing valuable insights into diabetes risk during pregnancy and supporting healthcare practitioners in delivering timely and targeted care.

8.0 CLIENT FEEDBACK

Harchana & Malavika:

- successfully completed all Python programming following the progress given.

- Does not include an explanation of the dataset & features involved.

- a bit lacking in discussion of results.

- have the potential to be polished further, with a more solid research methodology style to conduct projects.

9.0 Appendix

GitHub link:
https://github.com/NiesHW/SECB3203_P4B/tree/main/Group_Project/Group%2018

Pictures taken during the first client meeting (15/10/2023) :

Pictures taken after the final presentation (15/01/2024):