جامعة بيرزيت
BIRZEIT UNIVERSITY

FACULTY OF ENGINEERING AND TECHNOLOGY

DEPARTMENT OF COMPUTER ENGINEERING

Artificial Intelligence - ENCS434

Prepared by:

Leyan Burait #1211439

Haneen Odeh #1210716

Instructor: Dr. Adnan Yahya

Date:26/1/2024

Section: 1

Abstract:

Diabetes is a complex chronic disease that requires accurate classification and prompt diagnosis for effective management. This project aims to utilize machine learning techniques and the Weka program and python language to develop a robust classification and diagnostic system for diabetes. By employing a diverse dataset containing clinical and laboratory features, we preprocess and transform the data to extract meaningful information. We then utilize various machine learning algorithms available in the Weka program, including decision trees, and neural networks, to train and optimize our classification model. And decision tree, naïve bias, neural network and random forest using Python language. By evaluating the model's performance metrics, including accuracy, precision, recall, and F1-score, we demonstrate the effectiveness of our system in predicting diabetes onset and categorizing patients into relevant diabetes types. The implementation of this system using the Weka program (Also python language) provides a user-friendly interface and efficient tool for healthcare professionals to aid in early diabetes detection and personalized treatment strategies.

Table of Contents

1. INTRODUCTION

1.1 Machine learning

Machine learning is branch of artificial intelligent that concerns how the computer can learn and adapt new circumstances, there are many learning techniques based on the desired outcome from the technique and the input available at the training process.

1.2 Models Training

In this project, various machine learning techniques, including decision trees, random forest, and neural networks, are utilized to train and optimize a classification model by both the weka program and python language.

1. A Decision Tree: algorithm is employed as a powerful tool for constructing a tree-like model of decisions and their possible consequences. It allows for the representation of rules and relationships between variables in a structured manner. This algorithm is particularly useful for classification tasks where the data can be split into distinct categories.

2. Random Forest: The Random Forest algorithm is a versatile and widely used machine learning technique that operates by constructing multiple decision trees during training and outputting the class that is the mode of the classes (classification) or mean prediction (regression) of the individual trees.

3. A Naive Bayes classifier: It is a probabilistic machine learning model that's used for classification task. It is based on the Bayes theorem

4. A Neural Network: a type of deep learning algorithm, are also employed in this project. Neural networks are inspired by the structure and function of the human brain and consist of interconnected nodes or "neurons." They are capable of learning complex patterns and relationships in the data, making them suitable for classification tasks.

The classification model is trained and optimized using these techniques, taking into account the specific requirements and characteristics of the dataset. The performance of the model is evaluated using different metrics to evaluate its accuracy and effectiveness in classifying new data points.

By training and testing datasets using decision trees, random forest, neural networks, and naïve bias we will calculate accuracy, precision, recall &F1 score.

1.3 Data set

This project was implemented using 1 dataset for Weka program the link below to obtain data set for patients, some of whom have diabetes and some of whom do not have it.

Link:

1) Pore, N. (2023, August 23). *Healthcare diabetes dataset*. Kaggle. https://www.kaggle.com/datasets/nanditapore/healthcare-diabetes

For Python language we used 3 different datasets

1) Pore, N. (2023, August 23). *Healthcare diabetes dataset*. Kaggle. https://www.kaggle.com/datasets/nanditapore/healthcare-diabetes

2) ACUR, S. (2020a, March 21). *Diabetes*. Kaggle. https://www.kaggle.com/datasets/salihacur/diabetes

3) Darabi, P. K. (2023b, July 28). *Diabetes_dataset_with_18_features*. Kaggle. https://www.kaggle.com/datasets/pkdarabi/diabetes-dataset-with-18-features

2. Methodology

In this section, we will briefly summarize all the steps we took to complete our project to detect diabetes We followed the following steps:

1. We first read the raw data and performed all the preprocessing

2. All pre-processed data was rewritten with the extracted data into a CSV file.

3. Next, we decided which model we would practice on, and made sure we had a balanced approach data and then trained the model.

4. We stored all trained models for future use and new test samples.

3.TESTING AND RESULTS

3.1 Weka program

The dataset we used contains 2768 entries, we divided it 80% for training data, and 20% for testing data.

Before going in depth with the different outputs, we need to consider the following metrics:

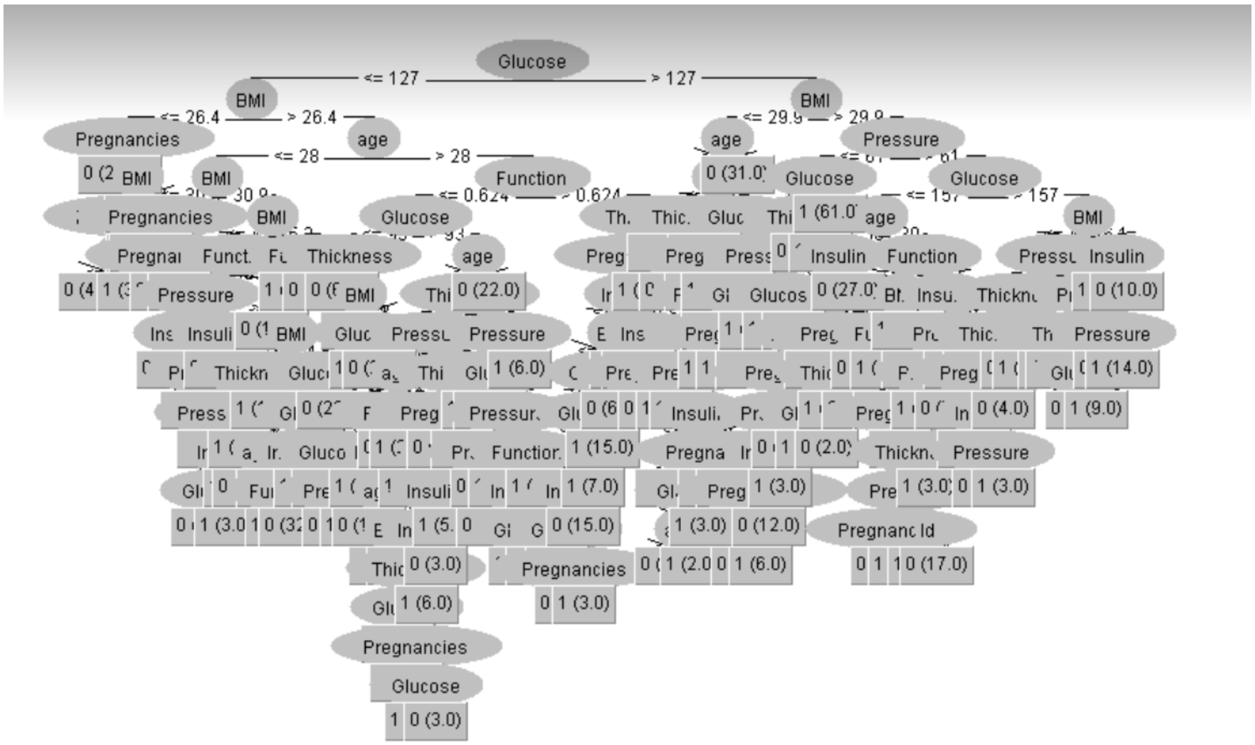
1. Precision: is the number of correctly classified positive examples divided by the total number of examples that are classified as positive.

2. Recall: is the number of correctly classified positive examples divided by the total number of actual positive examples in the test set.

3. Accuracy: is the number of correct classifications divided by the total number of test cases.

4. F1 score: is the harmonic mean of precision and recall, and it tends to closer the smaller of the two

3.1.1Desion tree



1.training

Time taken to build model: 0.16 see

Total Number of Instances: 2214

Correctly classified instances: 2210

In correctly classified instances 4

2.testing

Time taken to build model: 0.03 see

Total Number of Instances: 554

Correctly classified instances: 542

In correctly classified instances 12

Accuracy= correctly classified Instances/ Total number of instances in training set

Accuracy=542/554 = 0.978

TP=181 , FP= 10 , FN=2 , TN = 361

Precision= TP/TP+FP = 181/181+10 = 0.9476

Recall = TP/TP+FN=181/181+2 = 0.9891

F1 score= 2P*R/P+R = 0.96791

3.2Naive bais

1.traning

Total Number of Instances: 2214

Correctly classified instances: 1691

In correctly classified instances 523

2.test

Total Number of Instances: 554

Correctly classified instances: 422

In correctly classified instances 132

Accuracy= correctly classified Instances/ Total number of instances in data set

Accuracy=422/554 = 0.7617

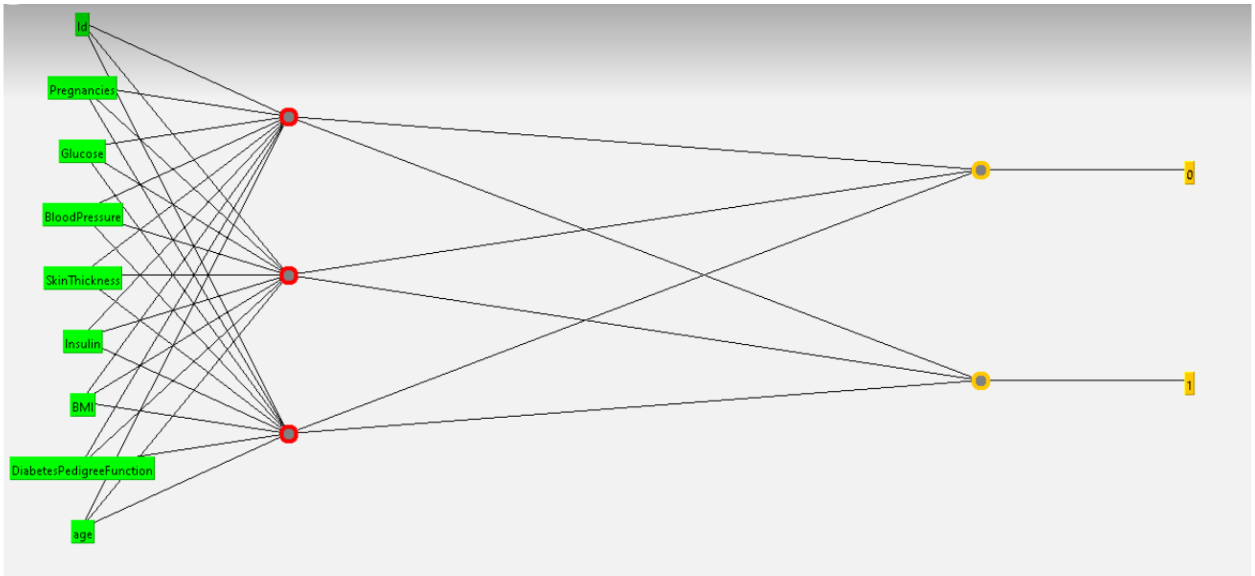
TP=117 , FP= 66 , FN=66 , TN = 305

Precision= TP/TP+FP = 117/554= 0.6393

Recall = TP/TP+FN=171/171+66 = 0.6393

F1 score= 2P*R/P+R = 0.6395

3.3 A Neural Network:



1.traning

Total Number of Instances: 2214

Correctly classified instances: 1825

In correctly classified instances 389

2.testing

Total Number of Instances: 554

Correctly classified instances: 455

In correctly classified instances 99

Accuracy= correctly classified Instances/ Total number of instances in data set

Accuracy=455/554 =0.8213

TP=128 , FP= 44 , FN=55 , TN = 327

Precision= TP/TP+FP = 128/128+44 = 0.7442

Recall = TP/TP+FN=128/128+55 = 0.6995

F1 score= 2P*R/P+R = 0.7211

3.2 Python Language

We used 3 different datasets with different number of entries to see the difference,

A short picture for Dataset1 as .csv

```
dataset1.csv > data
Pregnancies Glucose BloodPressure   SkinThickness   Insulin BMI DiabetesPedigreeFunction    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
```

A short picture for Dataset2 as .csv

```
dataset2.csv > data
Pregnancies Glucose BloodPressure   SkinThickness   Insulin BMI DiabetesPedigreeFunction    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
```

A short picture for Dataset3 as .csv

```
dataset3.csv > data
Age Gender  BMI SBP DBP FPG Chol    Tri HDL LDL ALT BUN CCR FFPG    smoking drinking    family_histroy
26  1   20.1    119 81  5.8 4.36    0.86    0.9 2.43    12  5.4 63.8    5.4 3   3   0   0
40  1   17.7    97  54  4.6 3.7 1.02    1.5 2.04    9.2 3.7 70.3    4.1 1   1   0   0
40  2   19.7    85  53  5.3 5.87    1.29    1.75    3.37    10.1    4.1 61.1    4.85    3   3   0   0
43  1   23.1    111 71  4.5 4.05    0.74    1.27    2.6 36.5    4.38    73.4    5.3 2   3   0   0
36  1   26.5    130 82  5.54    6.69    3.49    0.91    3.64    69.3    3.86    67.5    5.53    3   3
46  2   20.5    88  63  5.76    4.6 1   1.32    2.78    15  4.19    59  4.8 3   3   0   0
52  1   31.7    129 84  5.9 6.14    2.18    1.15    3.43    26  4.7 79  5.48    1   3   0   0
33  1   22.9    129 92  5.17    6.02    3.9 1.09    3.12    39.6    4.48    68.3    5.84    3   3   0
42  1   27.1    109 56  5.06    4.73    1.02    1.15    2.82    11.5    2.98    80.2    5.2 3   3   0
37  2   20.08   128 75  4.67    6.75    0.61    2.4 4.25    8.3 5.03    62.7    5.19    3   3   0   0
51  1   22  109 84  4.61    4.95    2.4 1.07    2.95    15.5    3.67    79.4    4.68    3   3   0   0
38  2   20.3    95  58  4.96    3.95    1.24    0.96    2.4 10.4    3   50.8    4.99    3   3   0   0
50  1   21.4    103 77  4.8 4.58    1.36    1.53    2.31    21.9    5.2 80  5.35    1   2   0   0
53  1   21.7    130 86  5.38    4.4 3.47    1.17    2.25    25.8    5.7 71  5.52    2   3   0   0
47  1   20.76   100 79  4.69    4.49    0.51    1.52    2.66    22.1    5.93    95.6    4.92    3   3
```

We had to install many packages to perform all the algorithms we used, such as

```python
main2.py > ...
#! Installing all packages needed
from sklearn.tree import DecisionTreeClassifier
from sklearn.naive_bayes import GaussianNB
from sklearn.neural_network import MLPClassifier
from sklearn.ensemble import RandomForestClassifier
from sklearn.model_selection import train_test_split
from sklearn.metrics import accuracy_score, precision_score, recall_score, f1_score
from sklearn.metrics import confusion_matrix
import pandas as pd
import time
```

At first, we had to make the program read all 3 datasets and let the user decide which dataset to use

```python

#! Load our datasets into DataFrames
data1 = pd.read_csv('dataset1.csv', delimiter='\t')
data2 = pd.read_csv('dataset2.csv', delimiter='\t')
data3 = pd.read_csv('dataset3.csv', delimiter='\t')

#! Letting the user decide which dataset he/she wants
print("Last ID for dataset1:2768")
print("Last ID for dataset2:768")
print("Last ID for dataset3:4303, Note:(This dataset has more number of features)")

```

Based on the user's dataset, we divided it 80% training set and 20% testing set

```python
#! Separate features (X) and labels (y), X for all features except Outcome,y for Outcome feature
X = data.drop('Outcome', axis=1)
y = data['Outcome']

#! Split the dataset into training and testing sets (80% training and 20% testing)
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state=42)

```

We let the user decides which algorithm he wants to perform by showing this menu

```python
while True:
    #! Ask the user to choose the algorithm
    print("\n\t")
    print("Choose the algorithm you want to test:")
    print("1. Decision Tree")
    print("2. Naive Bayes")
    print("3. Neural Network")
    print("4. Random Forest")
    print("5. Exit")
    choice = input("Enter your choice (1, 2, 3, 4, or 5): ")

```

We had to initialize each algorithm so that we can perform and run it later

```python

    if choice == '1':
        #! Initialize the decision tree model
        model = DecisionTreeClassifier()
    elif choice == '2':
        #! Initialize the Naive Bayes model
        model = GaussianNB()
    elif choice == '3':
        #! Initialize the Neural Network model
        model = MLPClassifier(random_state=1, max_iter=300)
    elif choice == '4':
        #! Initialize the Random Forest model
        model = RandomForestClassifier(n_estimators=100, random_state=42)
    elif choice == '5':
        break
```

After that we trained our model, made predictions on the test set, and calculated the confusion matrix

```python
    #! Train the model
    model.fit(X_train, y_train)

    #! Start time
    start_time = time.time()

    #! Make predictions on the test set
    y_pred = model.predict(X_test)

    #! Calculate confusion matrix
    conf_matrix = confusion_matrix(y_test, y_pred)

```

After that, we calculated the evaluation metrics (Precision, Accuracy, Recall and F1-score), and rounded each one to 5 decimal places

```python
#! Calculate evaluation metrics
accuracy = accuracy_score(y_test, y_pred)
precision = precision_score(y_test, y_pred)
recall = recall_score(y_test, y_pred)
f1 = f1_score(y_test, y_pred)

#! Round the evaluation metrics to 5 decimal places
accuracy = round(accuracy, 5)
precision = round(precision, 5)
recall = round(recall, 5)
f1 = round(f1, 5)
```

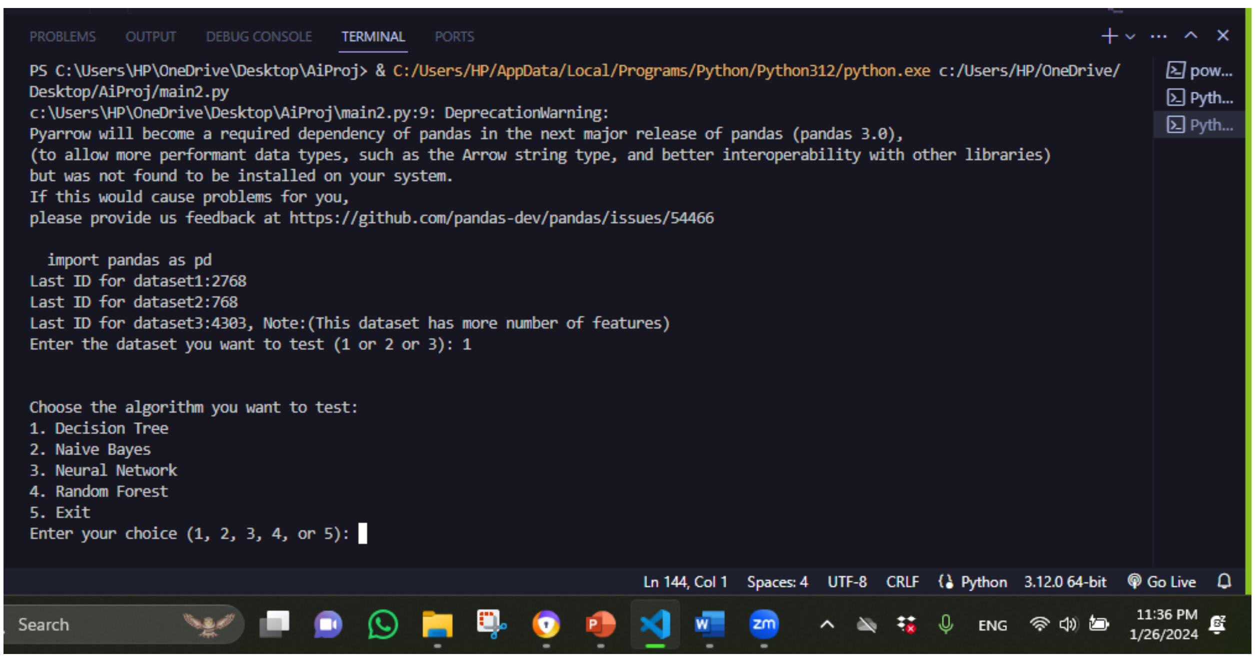
And finally, printing all the results to the screen

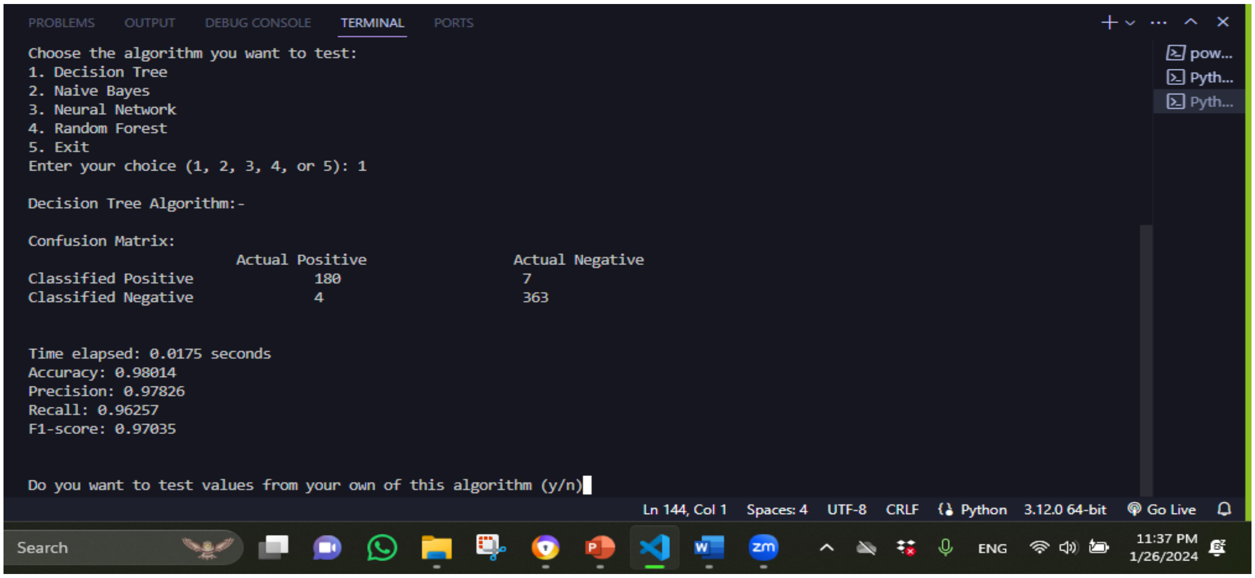
```python
#! Print confusion matrix
print("Confusion Matrix:")
print("\t\t\tActual Positive\t\t\tActual Negative")
print("Classified Positive\t\t", conf_matrix[1, 1], "\t\t\t", conf_matrix[1, 0])
print("Classified Negative\t\t", conf_matrix[0, 1], "\t\t\t", conf_matrix[0, 0])

#! Print evaluation metrics
print("\n")
print("Time elapsed:", time_elapsed, "seconds")
print("Accuracy:", accuracy)
print("Precision:", precision)
print("Recall:", recall)
print("F1-score:", f1)
print("\n")
```

The final step was to let the user enter his own values to see what will the output be for the diabetes prediction

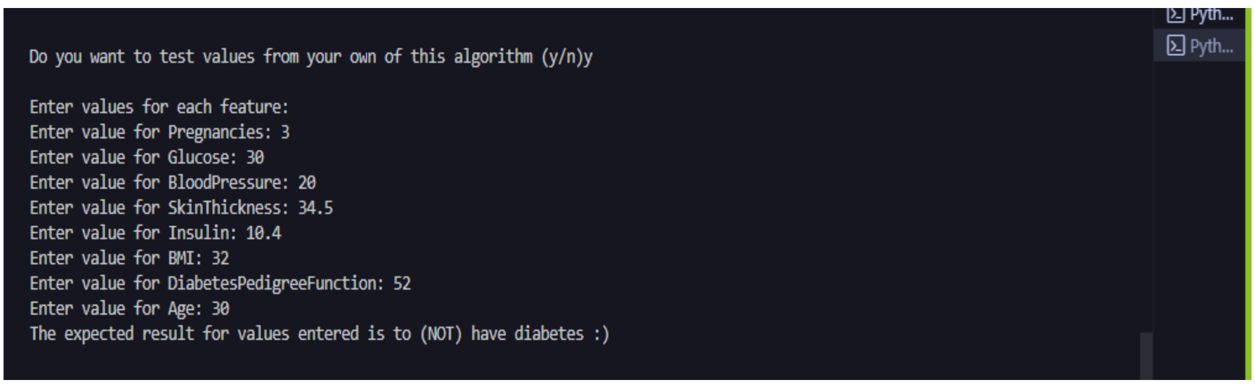
```python
#! Ask the user to enter values for each feature
value1 = input(f"Do you want to test values from your own of this algorithm (y/n)")
if value1.lower() == 'y':
    print("\nEnter values for each feature:")
    features = {}
    for column in X.columns:
        value = input(f"Enter value for {column}: ")
        features[column] = [value]

    #! Create a DataFrame with the user-provided values
    user_data = pd.DataFrame(features)

    #! Make predictions using the trained model
    user_pred = model.predict(user_data)

```

Some pictures for the results:-

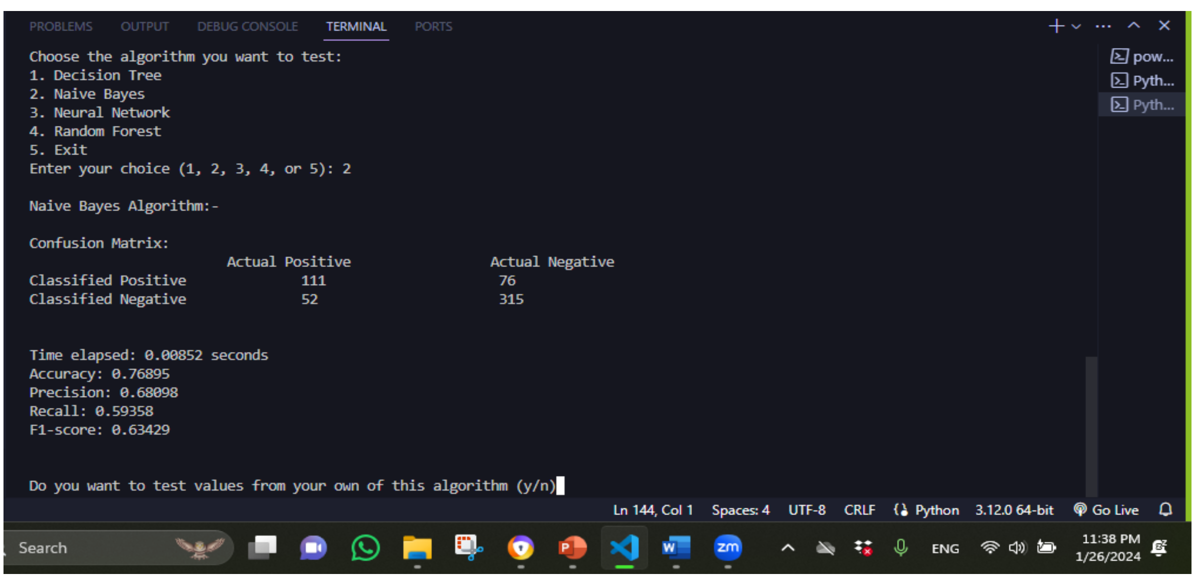




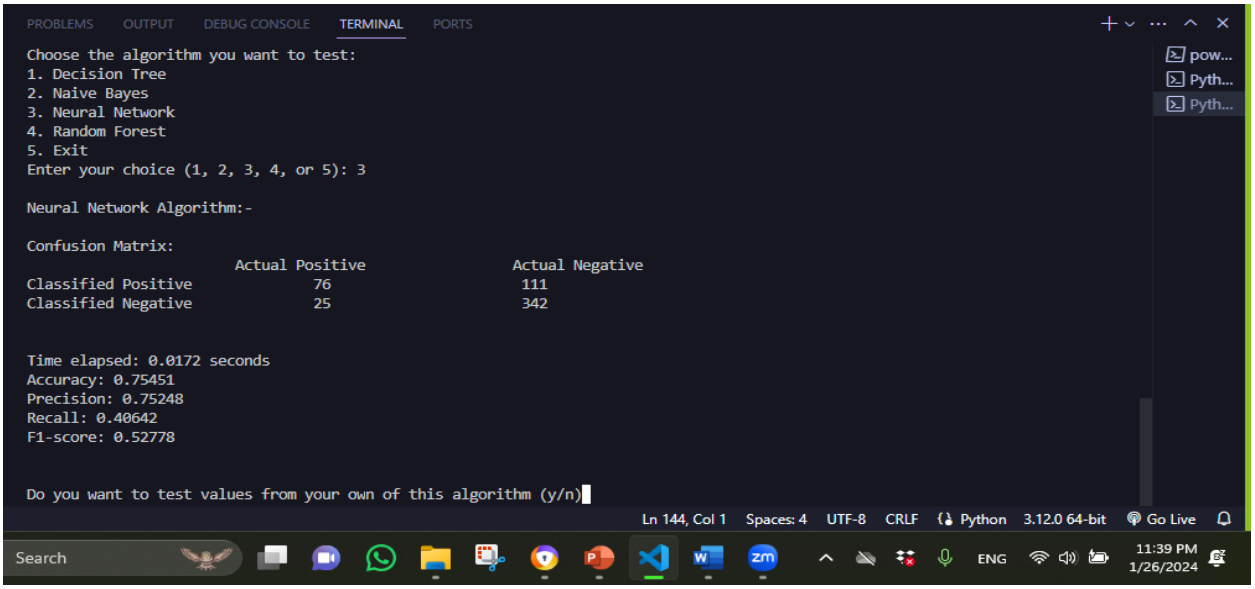
```
Choose the algorithm you want to test:
1. Decision Tree
2. Naive Bayes
3. Neural Network
4. Random Forest
5. Exit
Enter your choice (1, 2, 3, 4, or 5): 1

Decision Tree Algorithm:-

Confusion Matrix:
                        Actual Positive                 Actual Negative
Classified Positive          180                          7
Classified Negative          4                            363


Time elapsed: 0.0175 seconds
Accuracy: 0.98014
Precision: 0.97826
Recall: 0.96257
F1-score: 0.97035


Do you want to test values from your own of this algorithm (y/n)
```

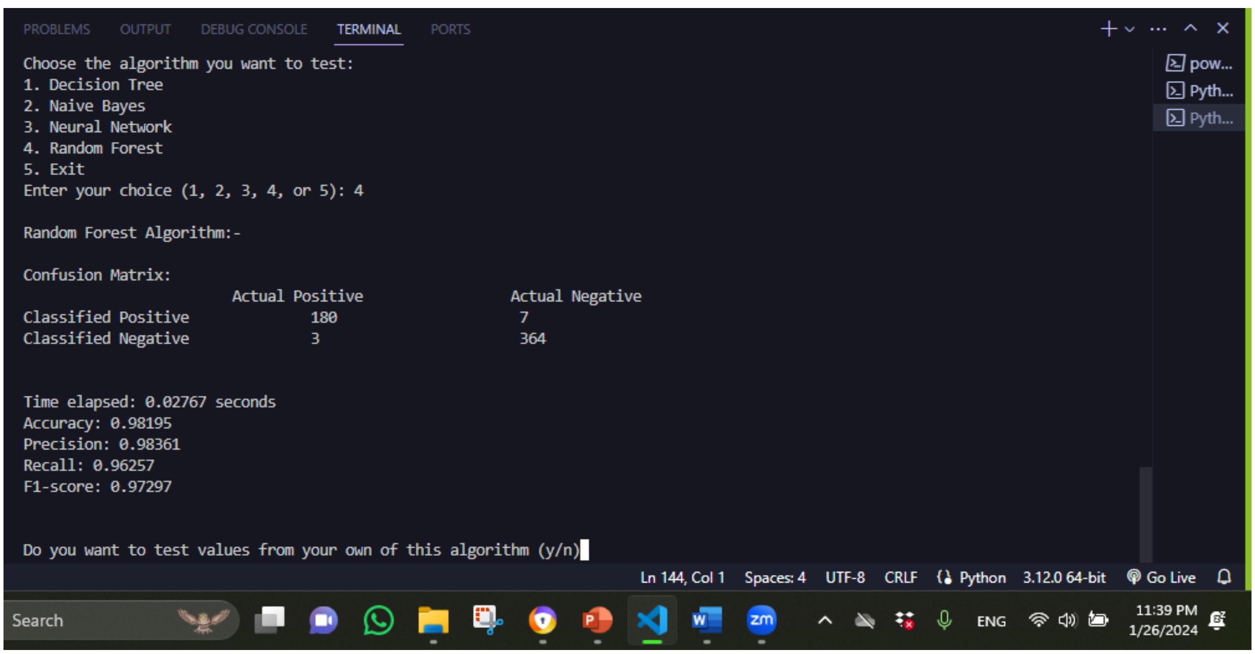



```
Do you want to test values from your own of this algorithm (y/n)y

Enter values for each feature:
Enter value for Pregnancies: 3
Enter value for Glucose: 30
Enter value for BloodPressure: 20
Enter value for SkinThickness: 34.5
Enter value for Insulin: 10.4
Enter value for BMI: 32
Enter value for DiabetesPedigreeFunction: 52
Enter value for Age: 30
The expected result for values entered is to (NOT) have diabetes :)
```

```
Choose the algorithm you want to test:
1. Decision Tree
2. Naive Bayes
3. Neural Network
4. Random Forest
5. Exit
Enter your choice (1, 2, 3, 4, or 5): 2

Naive Bayes Algorithm:-

Confusion Matrix:
                        Actual Positive                 Actual Negative
Classified Positive          111                          76
Classified Negative          52                           315


Time elapsed: 0.00852 seconds
Accuracy: 0.76895
Precision: 0.68098
Recall: 0.59358
F1-score: 0.63429


Do you want to test values from your own of this algorithm (y/n)
```


PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL PORTS
Choose the algorithm you want to test:
1. Decision Tree
2. Naive Bayes
3. Neural Network
4. Random Forest
5. Exit
Enter your choice (1, 2, 3, 4, or 5): 3
Neural Network Algorithm:-
Confusion Matrix:
Actual Positive Actual Negative
Classified Positive 76 111
Classified Negative 25 342
Time elapsed: 0.0172 seconds
Accuracy: 0.75451
Precision: 0.75248
Recall: 0.40642
F1-score: 0.52778
Do you want to test values from your own of this algorithm (y/n)
Ln 144, Col 1 Spaces: 4 UTF-8 CRLF Python 3.12.0 64-bit Go Live
Search
ENG
11:39 PM
1/26/2024


PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL PORTS
Choose the algorithm you want to test:
1. Decision Tree
2. Naive Bayes
3. Neural Network
4. Random Forest
5. Exit
Enter your choice (1, 2, 3, 4, or 5): 4
Random Forest Algorithm:-
Confusion Matrix:
Actual Positive Actual Negative
Classified Positive 180 7
Classified Negative 3 364
Time elapsed: 0.02767 seconds
Accuracy: 0.98195
Precision: 0.98361
Recall: 0.96257
F1-score: 0.97297
Do you want to test values from your own of this algorithm (y/n)
Ln 144, Col 1 Spaces: 4 UTF-8 CRLF Python 3.12.0 64-bit Go Live
Search
ENG
11:39 PM
1/26/2024