Faculty of Engineering and Technology

Electrical and Computer Engineering Department

First Semester 2024/2025

ENCS3340

Artificial Intelligence

Predictive Medical Diagnosis of Diabetes Based on Patient Symptoms Using Machine Learning

Prepared by:

Fatima Dawabsheh 1210827

Shahed Shrateh 1210444

Raghed Isleem 1211326

Instructor: Dr. Adnan Yahya

Section: 2

Date: 2/1/ 2025

***Abstract*— This project develops a machine learning-based system for diabetes classification and diagnosis, utilizing Python in Virtual Studio and Weka applications. We employ diverse clinical data, perform preprocessing and feature extraction, and train classification models using algorithms like decision trees, neural networks, and Naive Bayes. By evaluating performance metrics (accuracy, precision, recall, F1-score), we demonstrate the system's effectiveness in predicting diabetes and classifying patients. This implementation provides healthcare professionals with an efficient tool for early detection and personalized treatment planning.**

I. Introduction

- Machine Learning Overview

Machine learning also known as source, is a branch of artificial intelligence AI that deals with the possibility of the computer to learn. ML algorithms depends on the target output, and the input used for training.

- Model Development

This project also incorporates and discusses several machine learning models such as decision trees, random forests, Naive Bayes classifiers and even neural networks. The two development tools that are both used in development are Weka software and Python.

- Algorithm Description

•Decision Tree Classifier: Used to establish a decision tree which is preferable for classification tasks and represents choices and their results.

•Random Forest Classifier: The use of multiple decision trees in a way that optimizes a result function to obtain improved regression and classification predictions with lower variance.

•Naive Bayes Classifier: A derivative of Bayes' Theorem Probability Model that largely applies in classification challenges.

•Neural Network Classifier: A kind of a subcategory of the Deep Learning approach that attempts to mimic the functionality of the human brain. Due to the ability of learning complex patterns in the data, these models are appropriate for classification.

Evaluating the performance of models involves the use of accuracy, precision, recall and the F1-score.

II. Procedure

A. Datasets

We obtained two different data sets from kaggle which are given below. The first dataset had 2768 entries as processed [1].and the second had 4304 entries [2]. each dataset is saved as .csv file. Initial data cleaning was performed, which included removing the ID column from the first dataset. we used some common classifiers for use in classification problems such as decision trees, random forests, naive Bayes, and neural networks. We had ensured that our data was clean and was fit for use in the machine learning models before training the machine. The finally selected algorithms were applied to build classification models to classify the existence of diabetes. We implemented this by using Weak and by using the scikit- learn Python library, which has optimized machine learning algorithms.

III. . Performance Metrics:

We used the following metrics to assess the models performance:

-Precision: As a measure of the model's ability to predict the positive cases correctly, defined as the total number of true positives divided by the total of all the positive predictions of the model.

-Recall: Able to measure the totality of the True Positives discovered by a model as a fraction of the total count of actual true positives.

-Accuracy: To summarize, the overall correctness of our predictions, defined as the ratio of the numbers of properly classified data points to the total number of data points within the test set.

-F1 Score: Known as the harmonic mean of precision and recall, providing a balanced metric that is especially useful when classes are imbalanced.

- *TESTING AND RESULTS*

1. Weka program

Initially, we evaluated dataset one using Weka, the results:-

A. Decision tree

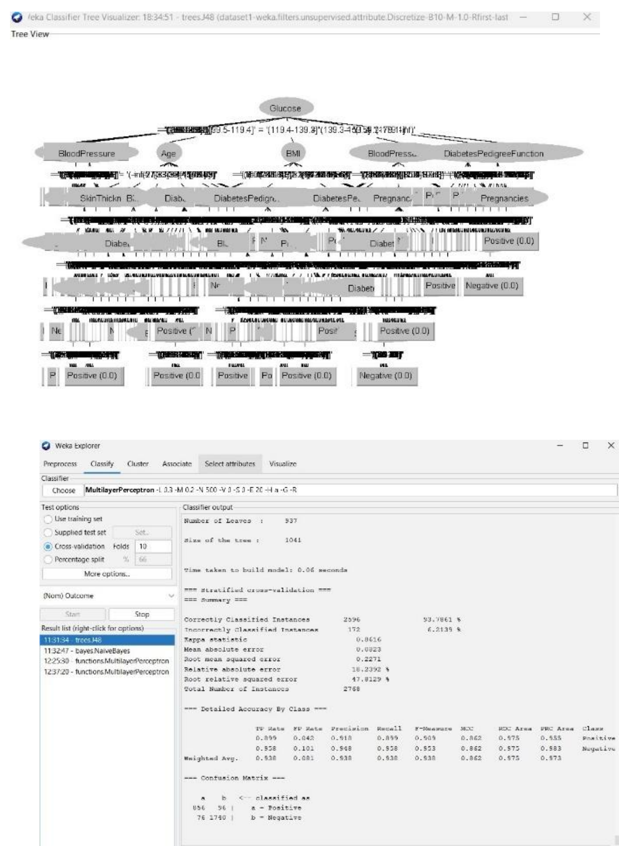

Figure 1:- Decision tree for dataset 1

The decision tree model's results showed a correctly classified instance rate of 93.78%, with a precision, recall, and F-measure equal to 0.938. The confusion matrix resulted in 856 true positives (TP), 96 false negatives (FN), 76 false positives (FP), and 1740 true negatives (TN). Based on these numbers, the overall accuracy, calculated as (TP+TN)/(TP+FP+TN+FN), was found to be 0.9378.

B. *Naïve Bayes*

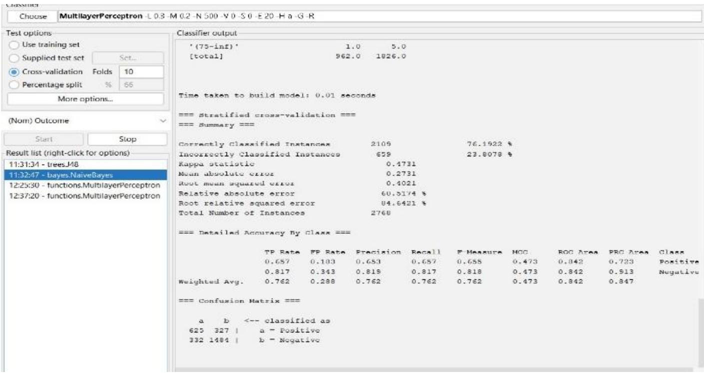

Figure 2:- Naïve Bayes for dataset 1

The *Naïve Bayes* model's results showed a correctly classified instance rate of 76.192%, with a precision, recall, and F-measure equal to 0.762. The confusion matrix resulted in 625 true positives (TP), 327 false negatives (FN), 332 false positives (FP), and 1484 true negatives (TN). Based on these numbers, the overall accuracy, calculated as (TP+TN)/(TP+FP+TN+FN), was found to be 0.7619.

C. *Neural network*

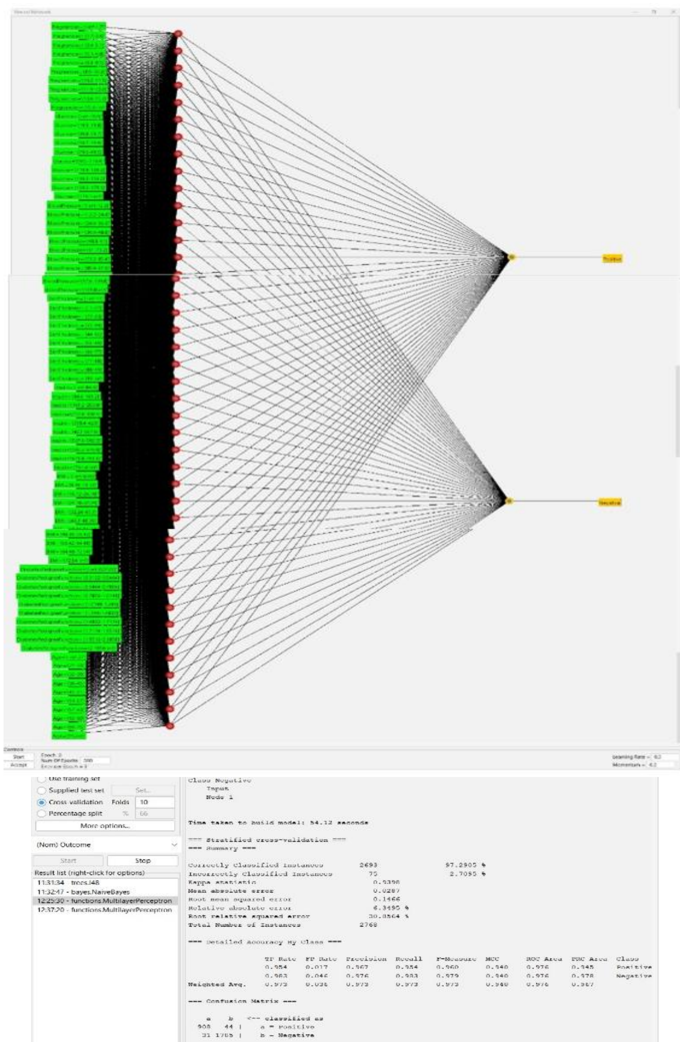

Figure 4:- Neural network for dataset 1

The *Neural network* model's results showed a correctly classified instance rate of 97.29%, with a precision, recall, and F-measure equal to 0.973. The confusion matrix resulted in 908 true positives (TP), 44 false negatives (FN), 31 false positives (FP), and 1785 true negatives (TN). Based on these numbers, the overall accuracy, calculated as (TP+TN)/(TP+FP+TN+FN), was found to be 0.9729.

D. *random forest*

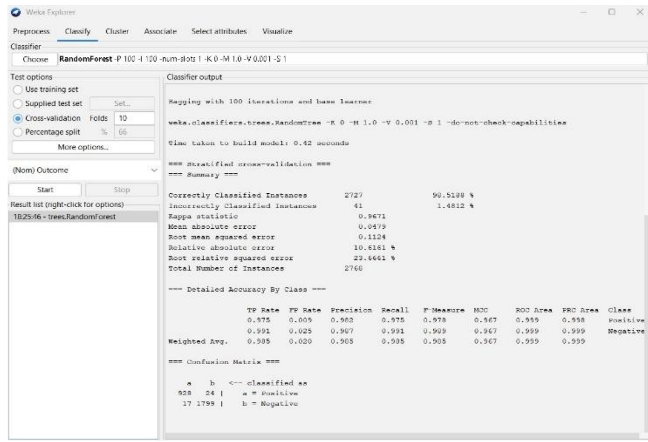

Figure 5:- random forest for dataset1

The *random forest* model's results showed a correctly classified instance rate of 98.518% with a precision, recall, and F-measure equal to 0.985. The confusion matrix resulted in 928 true positives (TP), 24 false negatives (FN), 17 false positives (FP), and 1799 true negatives (TN). Based on these numbers, the overall accuracy, calculated as (TP+TN)/(TP+FP+TN+FN), was found to be 0.9851.

- *Dataset 2:-*

A. Decision tree

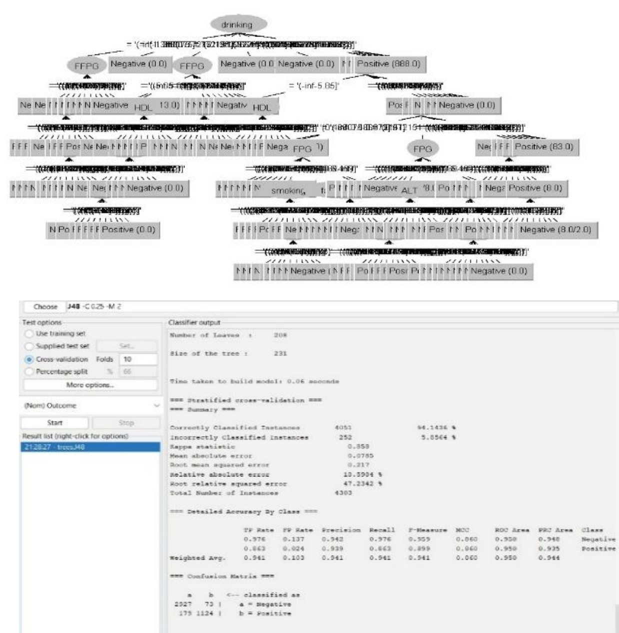

Figure 3:- decision tree for dataset 2

The decision tree model's results showed a correctly classified instance rate of 94.14, with a precision, recall, and F-measure equal to 0.941. The confusion matrix resulted in 2927 true positives (TP), 73 false negatives (FN), 179 false positives (FP), and 1124 true negatives (TN). Based on these numbers, the overall accuracy, calculated as (TP+TN)/(TP+FP+TN+FN), was found to be 0.9414.

B. Naïve Bayes

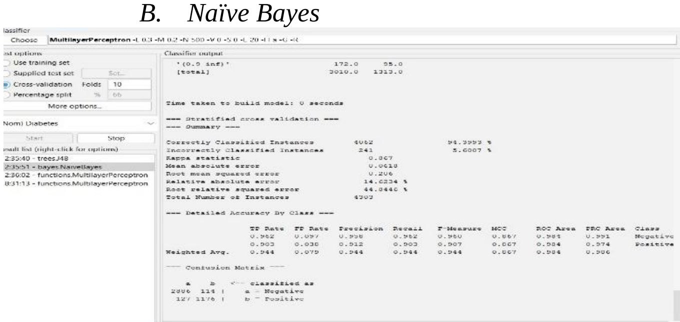

Figure 6:- Naïve Bayes for dataset 2

The *Naïve Bayes* model's results showed a correctly classified instance rate of 94.39%, with a precision, recall, and F-measure equal to 0.944. The confusion matrix resulted in 2886 true positives (TP), 114 false negatives (FN), 127 false positives (FP), and 1176 true negatives (TN). Based on these numbers, the overall accuracy, calculated as (TP+TN)/(TP+FP+TN+FN), was found to be 0.9439.

C. Neural network

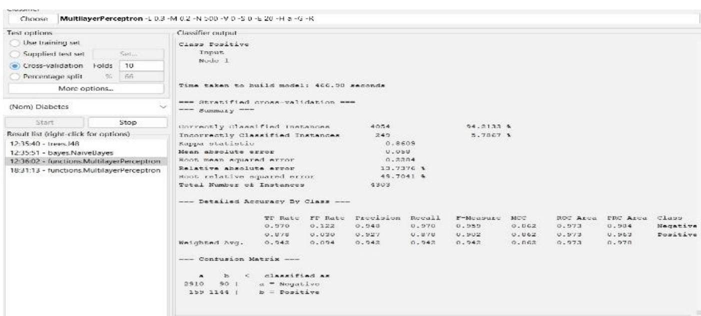

Figure 7:- Neural network for dataset 2

The Neural network model's results showed a correctly classified instance rate of 94.213%, with a precision, recall, and F-measure equal to 0.942. The confusion matrix resulted in 2910 true positives (TP), 90 false negatives (FN), 159 false positives (FP), and 1144 true negatives (TN). Based on these numbers, the overall accuracy, calculated as (TP+TN)/(TP+FP+TN+FN), was found to be 0.9421.

D. random forest

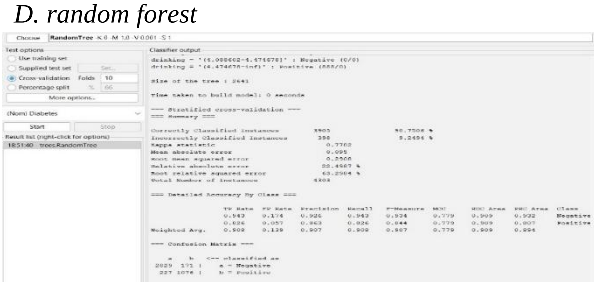

Figure 10:- random forest for dataset 2

The *random forest* model's results showed a correctly classified instance rate of 90.750%, with a precision, recall, and F-measure equal to 0.907. The confusion matrix resulted in 2829 true positives (TP), 171 false negatives (FN), 227 false positives (FP), and 1076 true negatives (TN). Based on these numbers, the overall accuracy, calculated as (TP+TN)/(TP+FP+TN+FN), was found to be 0.9075.

Based on our results, the random forest algorithm achieved the highest accuracy on dataset one at 98.518%. For dataset two, the naive Bayes algorithm was most effective, reaching an accuracy of 94.39%.

2. A Web Interface for Diabetes Diagnosis

We created a user website interface to make our project more practical and accessible, enabling users to test for a possible diabetes diagnosis

Figure 11:- Page 1

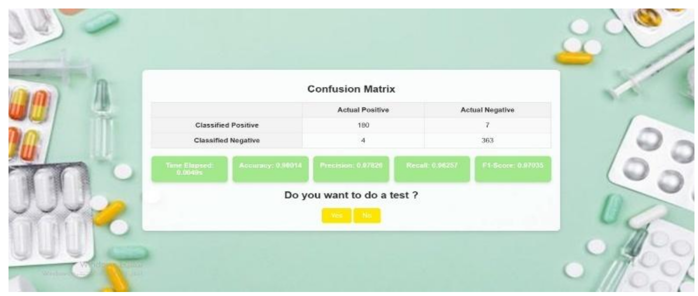

Figure 12:- Page 2

Figure 9:-Page 3

Figure 8:- Page 4

3. Python code

```python
# 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,confusion_matrix
import time
import pandas as pd

# Load datasets into DataFrames
Dataset1 = pd.read_csv('dataset1.csv',delimiter=',')
Dataset2 = pd.read_csv('dataset2.csv',delimiter=',')

# target column
t_columns = {
    '1': 'Outcome',
    '2': 'Diabetes'
}

print("Dataset1 : 2768")
print("Dataset2 : 4303 , This dataset has more number of features")
choice = input("Enter the dataset you want to test 1 or 2 : ")
if choice == '1':
    data = Dataset1
elif choice == '2':
    data = Dataset2
else:
    print("Invalid dataset")
    exit()
```

Figure 13:- code 1

We import the necessary libraries, such as sklearn for machine learning models and metrics, and pandas for handling datasets. And to make the data ready for analysis we import the two files as dataset 1 and dataset 2 from local CSV files using the method pd.read_csv().

```python
# Verify the target column exists
target_column = t_columns.get(choice)
if target_column not in data.columns:
    print(f"Error: The'{target_column}'column is missing in the selected dataset")
    print("Available columns:", data.columns)
    exit()

# Separate features (X) and labels (y)
X = data.drop(target_column, axis=1)
y = data[target_column]
# Split the dataset into 80% training and 20% testing sets
Xtrain, Xtest, ytrain, ytest = train_test_split(X, y, test_size=0.2, random_state=42)
```

Figure 14:- code 2

The target column name is obtained by using the choice of the user as the key for accessing the value in the dictionary, t_columns. If the given target column is not in the selected dataset, we display which columns are available and then the program ceases operation.

```python
while True:
    print("\t")
    print("Choose the algorithm :")
    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: ")

    if Choice == '1':
        model = DecisionTreeClassifier()
    elif Choice == '2':
        model = GaussianNB()
    elif Choice == '3':
        model = MLPClassifier(random_state=1, max_iter=300)
    elif Choice == '4':
        model = RandomForestClassifier(n_estimators=100, random_state=42)
    elif Choice == '5':
        break
    else:
        print("Invalid choice")
        continue
```

Figure 15:- code 3

A while True loop ensures the user can repeatedly choose an algorithm until they decide to exit and Based on the user's input, the appropriate machine learning model is instantiated

```python
# Train the model
model.fit(Xtrain, ytrain)

# Start time
start_time = time.time()

# Make predictions on the test set
ypred = model.predict(Xtest)

# Calculate confusion matrix
Conf_Matrix = confusion_matrix(ytest, ypred)

# Calculate evaluation metrics
Accuracy = accuracy_score(ytest, ypred)
precision = precision_score(ytest, ypred)
Recall = recall_score(ytest, ypred)
F1 = f1_score(ytest, ypred)

# Round to 5 decimal places
Accuracy = round(Accuracy,5)
precision = round(precision,5)
Recall = round(Recall,5)
F1 = round(F1,5)
Time_elapsed = round(time.time()-start_time,5)

# Print algorithm name
if choice == '1':
    print("\nDecision Tree Algorithm:")
elif choice == '2':
    print("\nNaive Bayes Algorithm:")
elif choice == '3':
    print("\nNeural Network Algorithm:")
elif choice == '4':
    print("\nRandom Forest Algorithm:")
```

Figure 16:- code 4

We trained the previously selected machine learning model using the training data (features in Xtrain and labels in ytrain). This was where the model learned the patterns in the data. And we recorded the current time before the prediction process started, used for measuring the time it takes to make a prediction. And we used the trained model to predict the labels (ypred) for the test dataset's features (Xtest).

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

# Test for user
value1 = input(f"Do you want to test values from your own of this algorithm (y/n)? ")
if value1.lower() =='y':
    features = {}
    for column in X.columns:
        value = input(f"Enter value for {column}: ")
        features[column]=[value]

    # Create a DataFrame
    User_Data = pd.DataFrame(features)

    # Make predictions
    User_Pred = model.predict(User_Data)

    # prediction
    if User_Pred[0] == 1:

        print("The expected result is to have diabetes")
    else:
        print("The expected result is not to have diabetes")
else:
    continue
```

Figure 17:- code 5

We ask the user if they want to input new values. And if the user choice yes, we prompt them to enter values for each column in the data. Then we create a new DataFrame using the values provided by the user. Then we use the trained model to predict the outcome based on the user input. And print whether the expected result indicates diabetes or not.

- ❖ Here is an example of what the output looks like when we run this code

```
Dataset1 : 2768
Dataset2 : 4303 , This dataset has more number of features
Enter the dataset you want to test 1 or 2 : 1

Choose the algorithm :
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 : 0.0 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)? y
Enter value for Pregnancies: 6
Enter value for Glucose: 148
Enter value for BloodPressure: 72
Enter value for SkinThickness: 35
Enter value for Insulin: 0
Enter value for BMI: 33.6
Enter value for DiabetesPedigreeFunction: 0.627
Enter value for Age: 50
The expected result is to have diabetes
```

Figure 18:- result 1

REFERENCES

[1] https://www.kaggle.com/datasets/salihacur/diabetes

[2] https://www.kaggle.com/datasets/pkdarabi/diabetes-dataset-with-18-features