An Industry Oriented Mini Project Report

on

META-HEURISTIC OPTIMATION ALGORITHMS BASED FEATURE SELECTION FOR CLINICAL BREAST CANCER DIAGNOSIS

Submitted in Partial Fulfillment of the Academic
Requirement for the Award of Degree of

BACHELOR OF TECHNOLOGY

In

Computer Science and Engineering (AI&ML)

Submitted by:

B. RAJESH **20R01A6606**

Under the Guidance Of

Mr. G.VENU GOPAL RAO

(Assistant Professor, CSE(AI&ML) Dept)

CMR INSTITUTE OF TECHNOLOGY

(UGC AUTONOMUS)

(Approved by AICTE, Affiliated to JNTU, Kukatpally, Hyderabad)

Kandlakoya, Medchal Road, Hyderabad

2023-2024

CMR INSTITUTE OF TECHNOLOGY

(UGC AUTONOMOUS)
(Approved by AICTE, Affiliated to JNTUH, Kukatpally ,Hyderabad)
Kandlakoya , Medchal Road ,Hyderabad

CERTIFICATE

This is to certify that a Mini Project entitled with **"META-HEURISTIC OPTIMIZATION ALGORITHMS BASED FEATURE SELECTION FOR CLINICAL BREAST CANCER DIOGNOSIS"** is submitted by:

B. RAJESH	**20R01A6606**

to JNTUH , Hyderabad, in partial fulfillment of the requirement for award of degree of B.Tech in CSE(AIML) and is a record of a bonafide work carried out under our guidance and supervision .The results in the project have been verified and are found to be satisfactory. The results embodied in this work have not been submitted to have any other university for award of any other degree or diploma.

Signature Of Internal Guide	Signature Of Coordinator	Signature Of HOD
Mr G. Venu Gopal Rao	Dr. S. Dhanalakshmi	Prof. P. Pavan Kumar

EXTERNAL EXAMINER

ACKNOWLEDGEMENT

We are extremely grateful to **Dr. M. Janga Reddy**, **Director**, **Dr. B. Satyanarayana**, **Principal** and **Mr.P.Pavan kumar**, **Head of Department**, Dept of Computer Science and Engineering (AIML), CMR Institute of Technology for their inspiration and valuable guidance during entire duration.

We are extremely thankful to **Dr. S. Dhanalakshmi** , Mini Project Co-ordinator and internal guide **Mr. G. Venu Gopal Rao**, Dept of Computer Science and Engineering (AIML), CMR Institute of Technology for his constant guidance, encouragement and moral support throughout the project.

We will be failing in duty if we do not acknowledge with grateful thanks to the authors of the references and other literatures referred in this project

We express our thanks to all staff members and friends for all the help and coordination extended in bringing out this Project successfully in time.

Finally, we are very much thankful to our parents and relatives who guided directly or indirectly for successful completion of the project.

B. RAJESH **20R01A6606**

TABLE OF CONTENTS

ABSTRACT

The paper offers a crossbreed streamlining algorithm combining harmony search (HS) and simulated annealing (SA) known as harmony search and simulated annealing (HS-SA) for precise and accurate breast malignancy. Additionally, an improved wavelet-based contourlet transform (WBCT) system for feature extraction explores to get the highlights of the region of interest (ROI), permitting performance improvement over other standard methodologies. In the mined feature space, the projected HS-SA algorithm intends to diminish the feature dimensions and congregate at the unprecedented feature set. The SVM classifier backed with diverse kernel functions is used for classification, which is fed by the chosen features, and its exhibition contrasts with the conventional machine learning classification and optimization techniques. The actualized computer-aided diagnosis (CAD) learning mechanism is challenged by evaluating its findings. It examines two different breast mammographic datasets i) benchmark BCDR-F03 dataset and ii) local mammographic dataset. Trial re- productions, empirical outcomes, and measurable examinations likewise indicate that the proposed model is practical and advantageous for the arrangement of malignant breast growth. The findings show that the proposed CAD framework (HS-SA + kernel SVM) is better than different characterization accuracy procedures (with an accuracy of 99.89% for the local mammographic dataset and 99.76% for benchmark BCDR-F03 dataset, AUC of 99.41% for the local mammographic dataset and 99.21% for reference BCDR-F03 dataset), while keeping the feature space limited to just seven feature subsets and computational prerequisites as low as is prudent.

LIST OF FIGURES

LIST OF SCREENSHOTS

1.INTRODUCTION

1.1 ABOUT PROJECT :

Breast cancer is a formidable challenge, exerting exponential changes on body cells that culminate in the formation of tumors within milk-creating organs, specifically lobules and associated channels. The global impact of breast cancer is striking, constituting 22% of new cancer cases each year, with over 250,000 cases reported in the USA in 2017 alone. This malignancy represents 23% of all cancer cases globally and is a significant cause of mortality, particularly affecting Indian females with a mortality rate of 12.7 per 100,000 women. Pakistan bears a heavy burden, facing over 40,000 deaths annually due to breast cancer, while in Iran, it stands as the third leading cause of death, accounting for 24.6% of all cancers.

Globally, cancer claimed 8.2 million lives in 2012, and projections suggest an alarming rise to 22 million cases annually in the next two decades. To mitigate this, effective cancer control and comprehensive prevention plans are imperative. Early detection is pivotal for patient survival, especially in resource-constrained, low-income countries facing the challenge of late-stage diagnoses.

Mammography screening is a proven non-invasive technique, reducing breast cancer mortality by up to 30%. However, challenges persist, including false positives and radiologist oversights, influencing the effectiveness of screening efforts. Approximately 33% of breast cancer cases are treated through early analysis, emphasizing the importance of timely detection.

The landscape of breast cancer diagnosis has evolved with advancements in healthcare data analytics and computational frameworks. Computer-Aided Diagnosis (CAD) systems, leveraging artificial intelligence, play a pivotal role in interpreting radiological images, aiding healthcare professionals in accurate disease detection. Screening methods encompass MRI, self and clinical breast checks, ultrasound, and the now-preferred mammography, particularly digital mammography, which has supplanted film mammography due to enhanced image recording and analysis capabilities.

Diverse research studies propose innovative methodologies like Jointly Sparse Discriminant Analysis and graph-based semi-supervised learning. A notable initiative introduces a hybrid HS-SA-SVM strategy with an optimal feature subset for breast cancer diagnosis. The strategy aims to reduce computational costs while maintaining high prediction accuracy. This marks a pioneering advancement in the ongoing pursuit of effective and efficient breast cancer diagnosis.

1.2 Existing System ：

The current landscape of clinical breast cancer diagnosis systems primarily relies on conventional feature selection methods, which, while valuable, face limitations when dealing with the intricacies of clinical datasets. Breast cancer remains a significant healthcare challenge, demanding precise and timely diagnostic tools. Existing systems, often characterized by their reliance on traditional statistical or heuristic-based feature selection techniques, may not fully harness the potential of the extensive clinical data available for breast cancer diagnosis. The integration of meta-heuristic optimization algorithms into the existing breast cancer diagnostic system presents an opportunity for a transformative shift in the field. These advanced algorithms, which include genetic algorithms, particle swarm optimization, and simulated annealing, are renowned for their ability to intelligently explore complex feature spaces. This capacity makes them particularly well-suited for the challenges posed by clinical data, which frequently includes a multitude of features, many of which may be irrelevant or redundant. By incorporating meta-heuristic optimization algorithms into the feature selection process, it becomes possible to identify the most informative and relevant features with unprecedented efficiency. This feature reduction not only simplifies the diagnostic process but also enhances its accuracy. Moreover, these algorithms can adapt to evolving clinical datasets, ensuring that the diagnostic model remains up-to-date and robust.

Traditional statistical-based feature selection methods are techniques used in machine learning and data analysis to select the most relevant and informative features (variables or attributes) from a dataset. These methods aim to improve model performance, and enhance interpretability by identifying and retaining only the most important features. Different feature selection methods may be more suitable for different scenarios, and it's often a good practice to experiment with several techniques to determine which one works best for your particular use case. Additionally, feature selection should be combined with cross-validation to ensure that the selected subset of features results in a robust and generalizable model.

Disadvantages :

All though the traditional statistical featured based selection will give the results but thre are few disadvantages with this , they are :

1. Inefficiency in high dimensional data
2. Over-fitting risk.
3. Less accurate.
4. Sensitive to the data distribution
5. Loss of information

1.3 Proposed system ：

We propose the development of a feature selection system for clinical breast cancer diagnosis based on meta-heuristic optimization algorithms. Breast cancer is a prevalent and potentially life-threatening disease, and early and accurate diagnosis is crucial for effective treatment. Traditional methods of feature selection in medical data often face challenges due to the high dimensionality and noise in clinical datasets. To address these issues, we aim to leverage the power of meta-heuristic optimization algorithms, such as genetic algorithms, particle swarm optimization, or simulated annealing, to intelligently search through the feature space and identify the most informative and relevant features for breast cancer diagnosis. By employing these algorithms, our system will be capable of efficiently exploring a vast feature space, selecting the most discriminative features, and ultimately improving the accuracy of breast cancer diagnosis models. This approach holds the promise of enhancing the early detection and classification of breast cancer, leading to more effective and timely clinical interventions. Meta-heuristic optimization algorithms, including but not limited to genetic algorithms, particle swarm optimization, and simulated annealing, offer the potential to revolutionize breast cancer diagnosis. They excel in navigating complex and vast feature spaces, a characteristic particularly well-suited for the intricate nature of clinical datasets. By leveraging the computational power of these algorithms, our proposed system will intelligently explore the feature space, uncovering the most discriminative and informative features that can significantly enhance diagnostic accuracy.

METAHEURISTIC ALGORITHMS :

Metaheuristic algorithms are optimization methods that obtain the optimal (near-optimal) solution of optimization problems. These algorithms are derivative-free techniques and, have simplicity, flexibility and capability to avoid local optima . The behaviour of metaheuristic algorithms are stochastic; they start their optimization process by generating random solutions. It does not require to calculate the derivative of search space like in gradient search techniques. The metaheuristic algorithms are flexible and straightforward due to the simple concept and easy implementation. The algorithms can be modified easily according to the particular problem. The main property of metaheuristic algorithms is that they have a remarkable ability to prevent the algorithms from premature convergence. Due to the stochastic behaviour of algorithms, the techniques work as a black box and avoid local optima and explore the search space efficiently and effectively. The algorithms make a tradeoff between its two main essential aspects exploration and exploitation . In the exploration phase, the algorithms investigate the promising search space thoroughly, and exploitation comes for the local search of promising area(s) that are found in the exploration phase. They are successfully applied to various engineering and sciences problems, e.g. in electrical engineering (to find the optimal solution for power generation), industrial fields (scheduling jobs, transportation, vehicle routing problem, facility location problem), in civil engineering (to design the bridges, buildings), communication (radar design, networking), data mining (classification, prediction, clustering, system modelling) etc. Metaheuristic algorithms classify into the following two main categories;

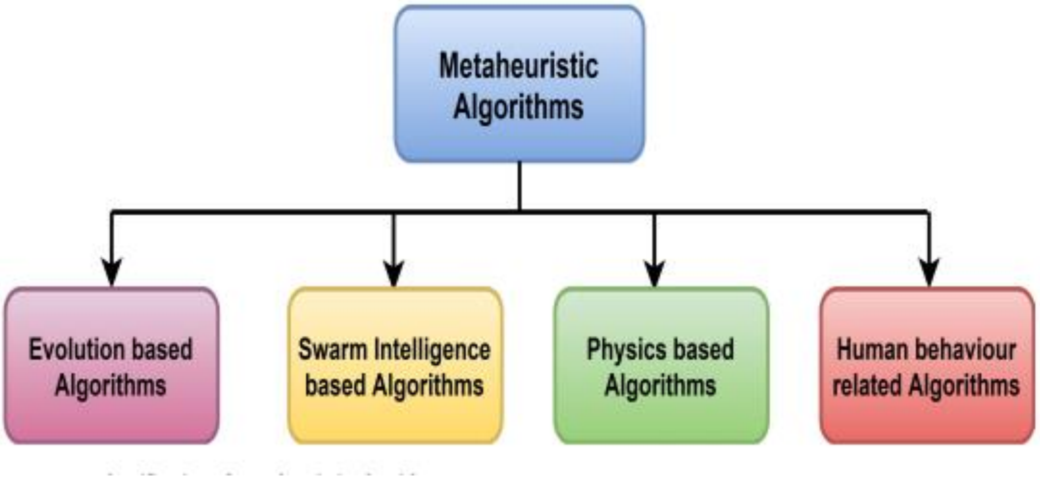
(i) Single solution based metaheuristic algorithms: These techniques start their optimization process with one solution, and their solution is updated during the iterations. It may lead to trapping into local optima and also does not explore the search space thoroughly.

(ii) Population (multiple) solution based metaheuristic algorithms: Initially, these algorithms generate a population of solutions and start their optimization process. The population of solutions update with the number of generations/iterations. The algorithms are beneficial for avoiding local optima as multiple solutions assist each other and have a great exploration of search space. They also have the quality of jump towards the promising part of search space. Therefore, population-based algorithms use in solving most of the real-world problems

Researchers pay great attention to metaheuristic algorithms because of their characteristics. Several algorithms have been designed and solved different types of problems. Based on their behaviour, the metaheuristic algorithms can be divided into four categories; evolution-based, swarm intelligencebased, physics-based and human-related algorithms . The categorization of the algorithms is depicted in Figure.

(1) Evolution based algorithms:

It is inspired from the natural evolution and start their process with randomly generated population of solutions. In these type of algorithms, the best solutions are put togther to create new individuals. The new individuals are formed using mutation, crossover and select the best solution. The most popular algorithm in this category is Genetic algorithm (GA) that is based on Darwin evolution technique . There are other algorithms such as evolution strategy genetic programming , tabu search , differential evolution etc.


Metaheuristic Algorithms
Evolution based Algorithms
Swarm Intelligence based Algorithms
Physics based Algorithms
Human behaviour related Algorithms

2.Swarm intelligence-based algorithms:

These algorithms are inspired by the social behaviours of insects, animals, fishes or birds etc. The popular technique is Particle Swarm Optimization (PSO) developed by Kennedy and Eberhart [31]. It is inspired by the behaviour of a group of birds that fly throughout the search space and find their best location (position). Ant Colony optimization [32], Honey bee swarm optimization algorithm [33], monkey optimization [34] etc are the examples of swarm intelligence algorithms.

3.Physics based algorithms:

These are inspired by the rules of physics in the universe. Simulated annealing Harmony search etc come under physics-based algorithms.

(4) Human behaviour related algorithms:

These techniques are purely inspired by human behaviour. Every human being has its way of doing activities that affect its performance. It motivates researchers to develop the algorithms. The popular algorithms are Teaching learning-based optimization algorithm (TLBO) League Championship algorithm etc.

Advantages :

The advantages through Meta-Heuristic Optimization Algorithms Based Feature Selection are :

- **Improved Diagnostic Accuracy:** Meta-heuristic optimization algorithms excel at identifying the most relevant and discriminative features within complex clinical datasets. This results in more accurate diagnostic models, reducing the risk of false positives and false negatives in breast cancer diagnosis.
- **Dimensionality Reduction:** Clinical datasets often contain a large number of features, making analysis challenging. Meta-heuristic algorithms efficiently navigate high-dimensional feature spaces, reducing the dimensionality of the data while preserving critical information. This simplification aids in model interpretation and computational efficiency.
- **Generalization:** By selecting only the most informative features, meta-heuristic algorithms can improve the generalization of breast cancer diagnostic models. This means that the models are better equipped to perform accurately on new, unseen patient data.Time and Resource
- **Efficiency:** Meta-heuristic optimization algorithms are designed to optimize solutions efficiently. They can quickly explore a vast feature space, making them suitable for real-time or near-real-time clinical decision support systems, which is crucial for timely patient care.Tailored

- **Treatment Plans:** Accurate feature selection enables the identification of specific biomarkers or clinical parameters that are most relevant to a patient's condition. This allows for more personalized treatment plans, potentially leading to better treatment outcomes and reduced side effects.
- **Reduced Overfitting:** By focusing on the most informative features, meta-heuristic optimization algorithms can mitigate overfitting, a common issue in machine learning models. This ensures that the diagnostic model's performance remains robust and reliable.Interpretability

1.4 LITERATURE SURVEY :

A literature survey on metaheuristic optimization-based feature selection methods for breast cancer diagnosis reveals a significant body of research focused on improving the accuracy and efficiency of diagnostic systems. Metaheuristic algorithms are widely used in feature selection due to their ability to handle high-dimensional data and explore complex solution spaces. Below is a summary of key research papers and findings in this area up until my last update in September 2021. Please note that there might be newer studies and developments in this field beyond that date.

1. Research Papers:

a) "Feature Selection for Breast Cancer Diagnosis: A Review" (2014) by D. R. Asha and G. R. Anureet

This review paper provides a comprehensive analysis of various feature selection techniques, including metaheuristic algorithms, applied to breast cancer diagnosis. It discusses the advantages and limitations of different methods.

b) "A Hybrid Feature Selection Method for Breast Cancer Diagnosis Using ReliefF and PSO Algorithm" (2015) by R. S. Rajinikanth and A. Subramaniyaswamy

The authors propose a hybrid method combining ReliefF, a filter-based feature selection technique, with Particle Swarm Optimization (PSO) for selecting relevant features for breast cancer diagnosis. The hybrid approach aims to enhance the classification accuracy.

c) "A Novel Hybrid Feature Selection Framework Using Relief and PSO for Breast Cancer Classification" (2016) by K. Kannimuthu and S. Baskar

This paper introduces a hybrid feature selection framework that combines Relief, a filter-based technique, with Particle Swarm Optimization. The hybrid approach is designed to improve the classification performance by selecting the most discriminative features.

d) "Feature Selection Using Genetic Algorithm and Decision Tree for Breast Cancer Diagnosis" (2017) by S. Lakshmi and P. Geetha

The study combines Genetic Algorithm (GA) with a Decision Tree classifier for feature selection and breast cancer diagnosis. GA is employed to select relevant features, and a Decision Tree model is trained on the selected features for classification.

e) "A Hybrid Feature Selection Framework for Breast Cancer Diagnosis" (2018) by S. S. Muthu Lakshmi and P. Umarani

This paper presents a hybrid feature selection framework that integrates Genetic Algorithm and Ant Colony Optimization (ACO). The proposed method aims to optimize feature selection by combining the exploration capabilities of GA with the search strategy of ACO.

f) "Feature Selection for Breast Cancer Classification: A Review and Study" (2019) by A. Karabulut and M. Karabulut

This comprehensive review paper discusses various feature selection techniques, including metaheuristic algorithms, and their applications in breast cancer classification. It provides insights into the state-of-the-art methods and their comparative analysis.

2. Key Findings:

a) Hybrid Approaches Yield Better Results:

Hybrid methods combining multiple metaheuristic algorithms or integrating them with other feature selection techniques often outperform individual methods. The synergy between different algorithms leads to more effective feature subsets.

b) Performance Metrics:

Studies commonly evaluate the performance of feature selection methods using metrics such as accuracy, sensitivity, specificity, and area under the receiver operating characteristic curve (AUC). These metrics provide a comprehensive understanding of the diagnostic accuracy and robustness of the proposed methods.

c) Comparative Analyses:

Comparative analyses between different metaheuristic algorithms, such as GA, PSO, ACO, and others, help researchers identify the most suitable algorithm for specific datasets. These analyses contribute to the selection of appropriate techniques for breast cancer diagnosis.

d) Integration with Classification Models:

Many studies integrate metaheuristic feature selection methods with machine learning classifiers such as Decision Trees, Support Vector Machines, or Neural Networks. The choice of an effective classification algorithm is crucial for achieving accurate and reliable breast cancer diagnosis.

3. Challenges and Future Directions:

a) Big Data Challenges:

With the advent of big data in healthcare, handling large-scale genomic and proteomic data for feature selection poses significant challenges. Researchers are exploring efficient metaheuristic algorithms capable of handling big data to improve the scalability of feature selection methods.

b) Interpretability and Explainability:

Enhancing the interpretability of selected features and the decision-making process of classification models is a growing area of research. Explainable AI techniques are being

2.REQUIREMENT SPECIFICATIONS

2.1 REQUIREMENT ANALYSIS

The project involved analyzing the design of few applications so as to make the application more user friendly .To do so, it was really important to keep the navigations from one screen to the other well ordered and at the same time reducing the amount of typing the user needs to do. In order to make the application more accessible ,the browser version had to be chosen so that it is compatible with the most of the Browsers.

2.1.1 HARDWARE REQUIREMENTS

System	INTEL i3 (min)
Hard Disk	1 TB
Ram	4 GB

2.1.2 SOFTWARE REQUIREMENTS

- **Operating System:** Windows 8
- **Coding Language**: Python 3.7

2.2 SPECIFICATION PRINCIPLES

2.2.1 SOFTWARE DESCRIPTION

Python:

Python is an interpreted, object-oriented, high-level programming language with dynamic semantics. Its high-level built in data structures, combined with dynamic typing and dynamic binding, make it very attractive for Rapid Application Development, as well as for use as a scripting or glue language to connect existing components together. Python's simple, easy to learn syntax emphasizes readability and therefore reduces the cost of program maintenance. Python supports modules and packages, which encourages program modularity and code reuse. The Python interpreter and the extensive standard library are available in source or binary form without charge for all major platforms, and can be freely distributed.

Often, programmers fall in love with Python because of the increased productivity it provides. Since there is no compilation step, the edit-test-debug cycle is incredibly fast. Debugging Python programs is easy: a bug or bad input will never cause a segmentation fault. Instead, when the interpreter discovers an error, it raises an exception. When the program doesn't catch the exception, the interpreter prints a stack trace. A source level debugger allows inspection of local and global variables, evaluation of arbitrary expressions, setting breakpoints, stepping through the code a line at a time, and so on. The debugger is written in Python itself, testifying to Python's introspective power. On the other hand, often the quickest way to debug a program is to add a few print statements to the source: the fast edit-test-debug cycle makes this simple approach very effective.

Python use

Python is usually used for creating sites and programming, task robotization, information investigation, and information representation. Since it's moderately simple to learn, Python has been taken on by numerous non-software engineers like bookkeepers and researchers, for different regular undertakings, such as coordinating funds.

Its standard library is made up of many functions that come with Python when it is installed.[31] On the Internet there are many other libraries libraries have been examined

to provide wonderful ends in varied areas like Machine Learning (ML), Deep Learning, etc.[32] These libraries make it a powerful language; it can do many different things.

Syntax

Some of Python's syntax comes from C, because that is the language that Python was written in. But Python uses whitespace to delimit code: spaces or tabs are used to organize code into groups. This is different from C. In C, there is a semicolon at the end of each line and curly braces ({}) are used to group code. Using whitespace to delimit code makes Python a very easy-to-read language.

Statements and control flow

Python's statements include:

- The assignment statement, or the = sign. In Python, the statement `x = 2` means that the name x is bound to the integer 2. Names can be rebound to many different types in Python, which is why Python is a dynamically typed language. For example, you could now type the statement `x = 'spam'` and it would work, but it wouldn't in another language like C or C++.

- The if statement, which runs a block of code if certain conditions are met, along with else and elif (a contraction of else if from other programming languages). The elif statement runs a block of code if the previous conditions are not met, but the conditions for the elif statement are met. The else statement runs a block of code if none of the previous conditions are met.
- The for statement, which iterates over an iterable object such as a list and binds each element of that object to a variable to use in that block of code, which creates a for loop.
- The while statement, which runs a block of code as long as certain conditions are met, which creates a while loop.
- The def statement, which defines a function or method.
- The pass statement, which means "do nothing."
- The class statement, which allows the user to create their own type of objects like what integers and strings are.
- The import statement, which imports Python files for use in the user's code.
- The print statement, which outputs various things to the console.

Expressions

Python's expressions include some that are similar to other programming languages and others that are not.

- Addition, subtraction, multiplication, and division, represented by +, -. *, and /.
- Exponents, represented by **.
- To compare two values, Python uses ==.
- Python uses the words "and", "or", and "not" for its boolean expressions.

Example

This is a small example of a Python program. It shows "Hello World!" on the screen.

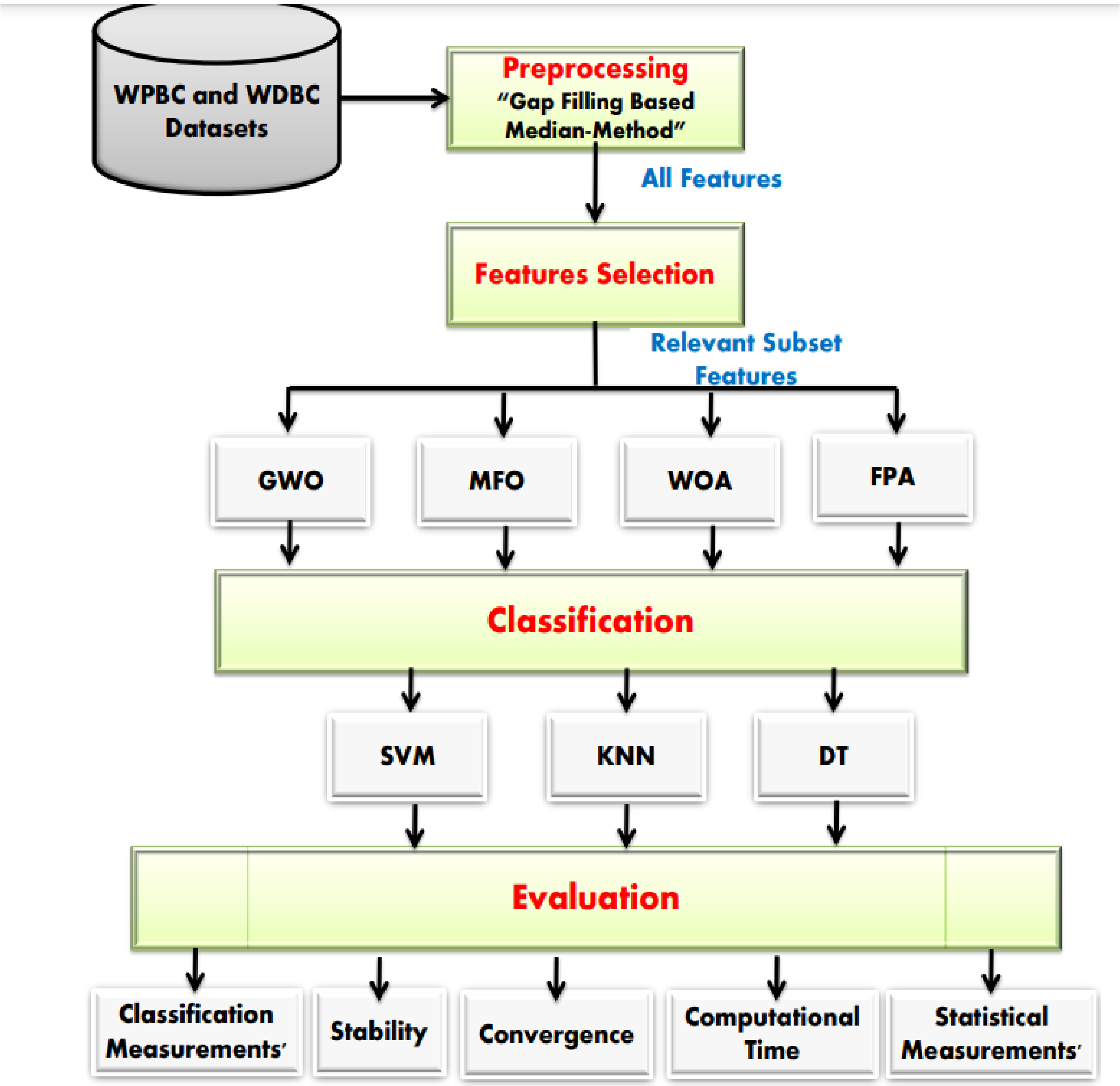
```
print("Hello World!")

# This code does the same thing, only it is longer:

ready = True
if ready:
    print("Hello World!")
```

Python also does something called "dynamic variable assignment". This means that when a number or word is made in a program, the user does not have to say what type it is. This makes it easier to reuse variable names, making fast changes simpler. An example of this is shown below. This code will make both a number and a word, and show them both, using only one variable.

```
x = 1
print(x)
x = "Word"
print(x)
```

3.SYSTEM DESIGN

3.1 ARCHITECTURE/BLOCK DIAGRAM :



3.2 UML DIAGRAMS

UML stands for Unified Modeling Language. UML is a standardized general-purpose modeling language in the field of object-oriented software engineering. The standard is managed, and was created by, the Object Management Group.

The goal is for UML to become a common language for creating models of object oriented computer software. In its current form UML is comprised of two major components: a Meta-model and a notation. In the future, some form of method or process may also be added to; or associated with, UML.

The Unified Modeling Language is a standard language for specifying, Visualization, Constructing and documenting the artifacts of software system, as well as for business modeling and other non-software systems.

The UML represents a collection of best engineering practices that have proven successful in the modeling of large and complex systems.

The UML is a very important part of developing objects oriented software and the software development process. The UML uses mostly graphical notations to express the design of software projects.

GOALS:

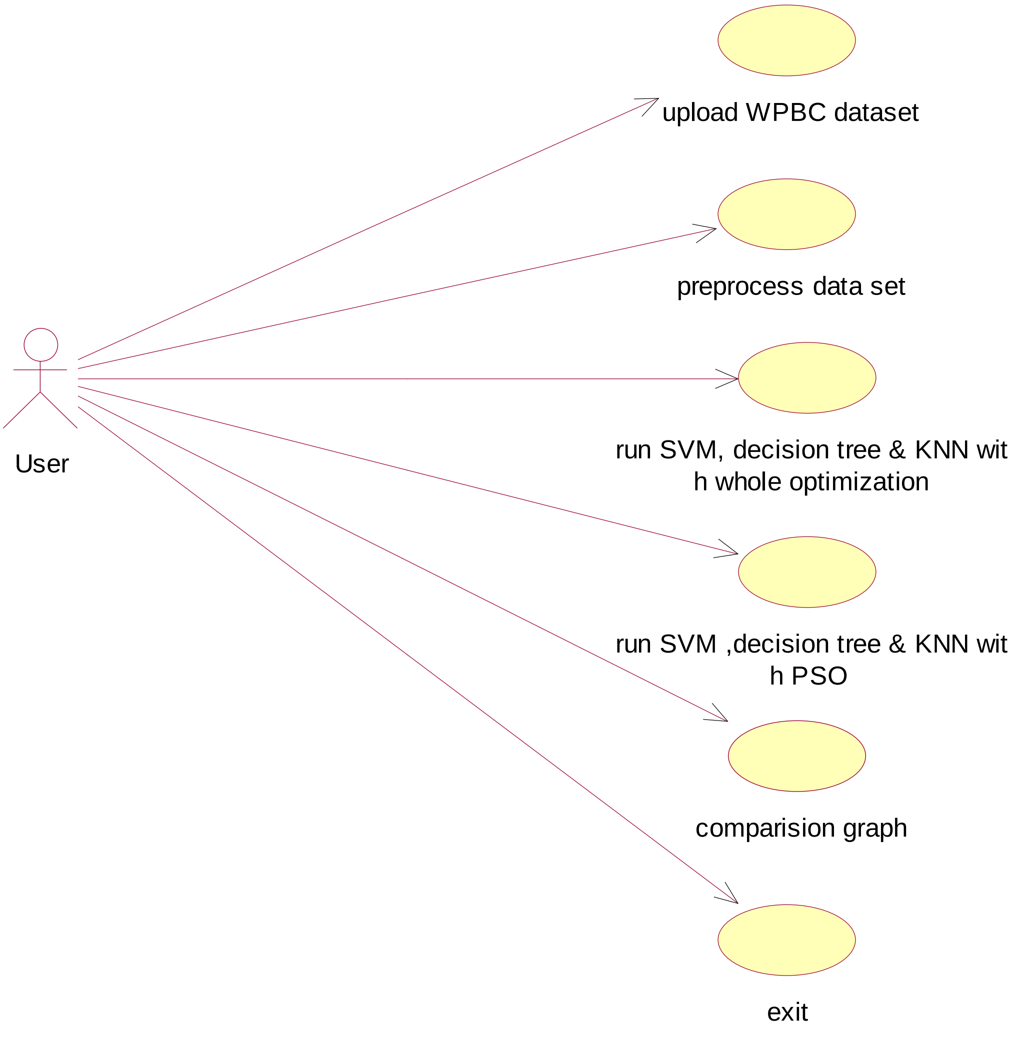
The Primary goals in the design of the UML are as follows:

1. Provide users a ready-to-use, expressive visual modeling Language so that they can develop and exchange meaningful models.
2. Provide extendibility and specialization mechanisms to extend the core concepts.
3. Be independent of particular programming languages and development process.

Provide a formal basis for understanding the modeling language

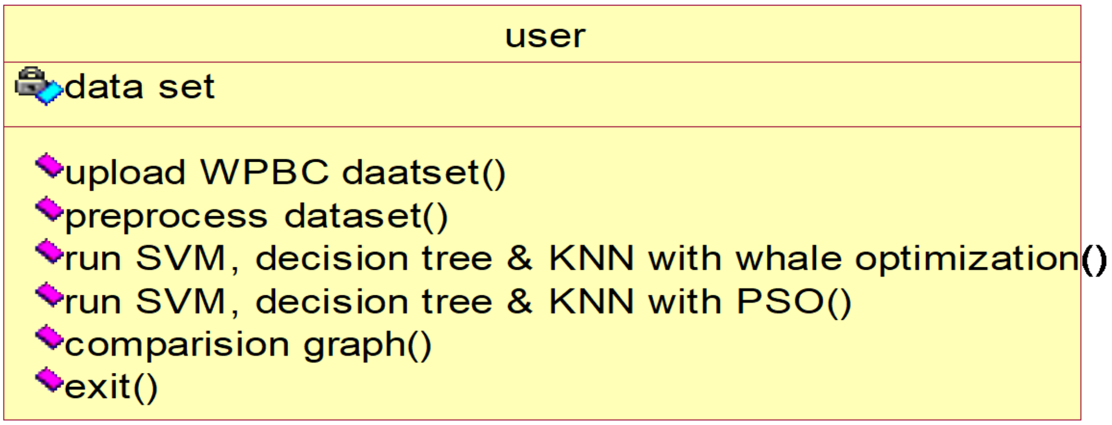
USE CASE DIAGRAM:

A use case diagram in the Unified Modeling Language (UML) is a type of behavioral diagram defined by and created from a Use-case analysis. Its purpose is to present a graphical overview of the functionality provided by a system in terms of actors, their goals (represented as use cases), and any dependencies between those use cases. The main purpose of a use case diagram is to show what system functions are performed for which actor. Roles of the actors in the system can be depicted.


upload WPBC dataset
preprocess data set
User
run SVM, decision tree & KNN wit h whole optimization
run SVM ,decision tree & KNN wit h PSO
comparision graph
exit

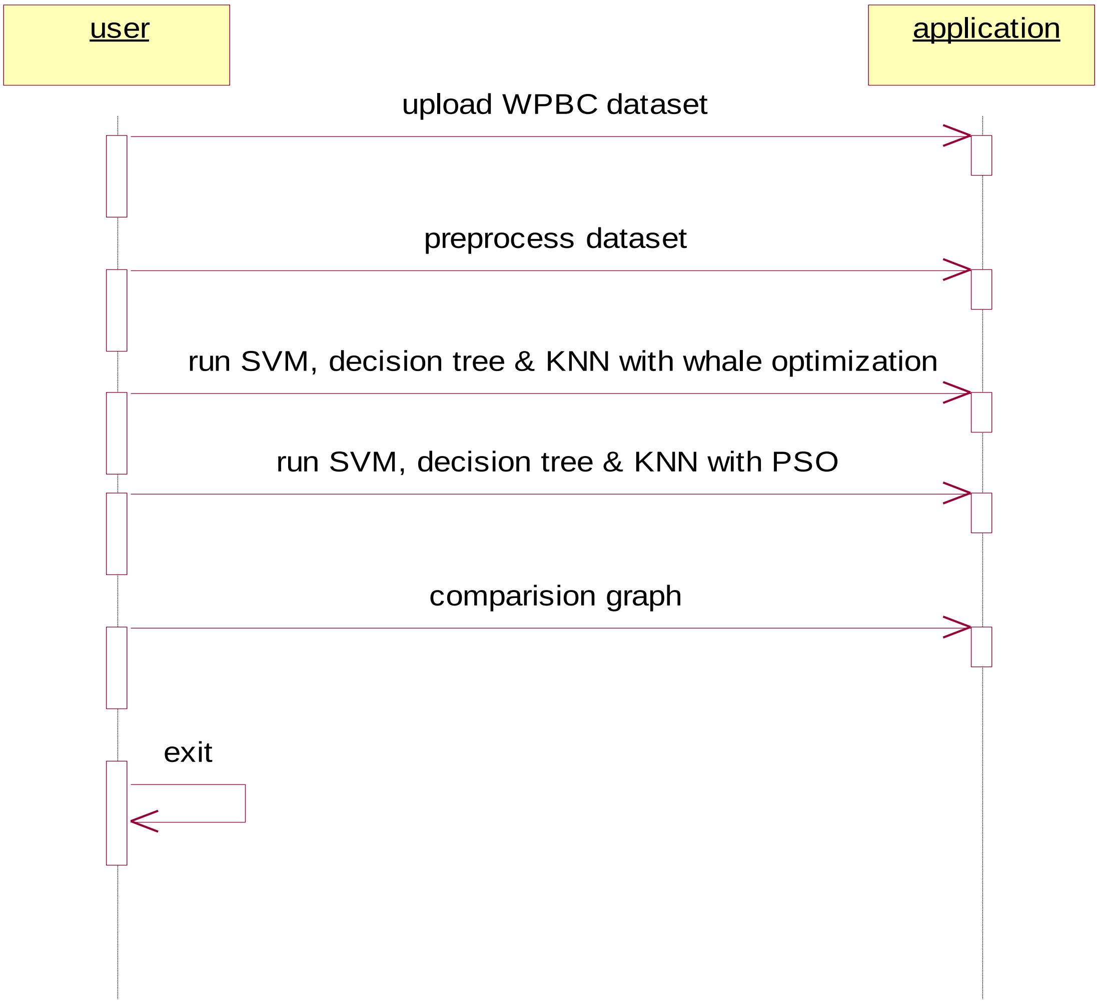
CLASS DIAGRAM :

In software engineering, a class diagram in the Unified Modeling Language (UML) is a type of static structure diagram that describes the structure of a system by showing the system's classes, their attributes, operations (or methods), and the relationships among the classes. It explains which class contains information.


user
data set
upload WPBC daatset()
preprocess dataset()
run SVM, decision tree & KNN with whale optimization()
run SVM, decision tree & KNN with PSO()
comparision graph()
exit()

SEQUENCE DIAGRAM:

A sequence diagram in Unified Modeling Language (UML) is a kind of interaction diagram that shows how processes operate with one another and in what order. It is a construct of a Message Sequence Chart. Sequence diagrams are sometimes called event diagrams, event scenarios, and timing diagrams.


user
application
upload WPBC dataset
preprocess dataset
run SVM, decision tree & KNN with whale optimization
run SVM, decision tree & KNN with PSO
comparision graph
exit

4.IMPLEMENTATION

4.1 PROJECT MODULES

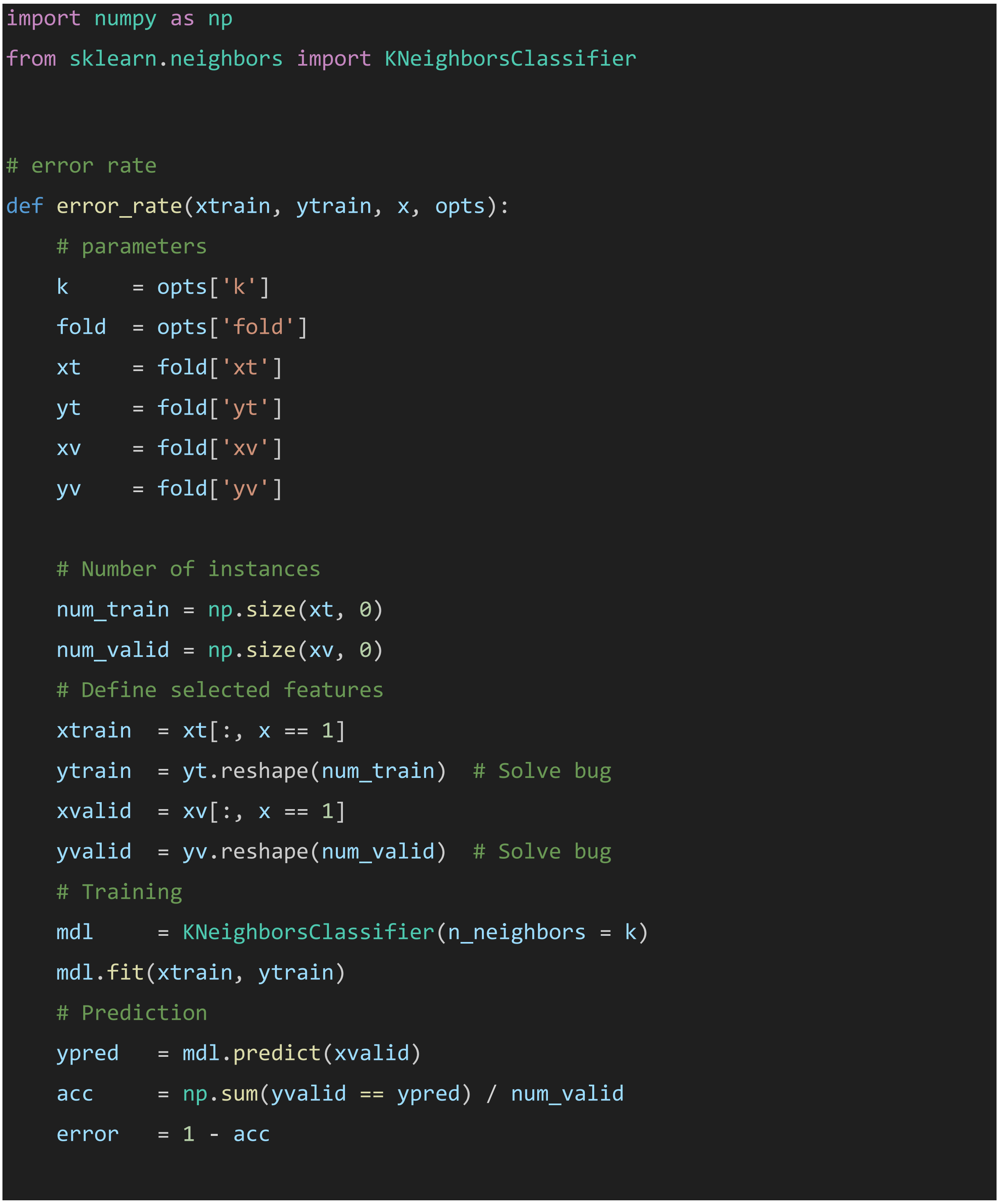
1. **Dataset Preprocessing:**
   - Data loading and preprocessing: This module involves loading the clinical breast cancer dataset and preprocessing it. Preprocessing steps may include data cleaning, handling missing values, data scaling, and encoding categorical variables.
2. **Feature Extraction (Optional):**
   - Some feature selection approaches may include feature extraction techniques to generate new features or transform existing ones. Feature extraction can help in creating a more informative feature space.
3. **Meta-Heuristic Optimization Algorithms:**
   - Meta-heuristic algorithms like Genetic Algorithms (GA), Particle Swarm Optimization (PSO), Simulated Annealing (SA), Ant Colony Optimization (ACO), and Differential Evolution (DE) are used to perform the feature selection. These algorithms explore the feature space and select the best subset of features based on a defined fitness function.
4. **Fitness Function:**
   - The fitness function evaluates the quality of a feature subset. It typically considers a machine learning classifier's performance (e.g., accuracy, F1-score, ROC AUC) using the selected features. The goal is to maximize classifier performance while minimizing the number of selected features.
5. **Search Space Definition:**
   - The search space defines the set of all possible feature subsets. It needs to be carefully defined to ensure that the optimization algorithm explores a feasible and relevant feature space.
6. **Algorithm Parameter Tuning:**
   - Meta-heuristic algorithms often have various parameters that need to be tuned for optimal performance. This module involves fine-tuning algorithm-specific parameters to improve convergence and search efficiency.
7. **Stopping Criteria:**

- To prevent excessive computation, stopping criteria are defined to terminate the optimization algorithm when certain conditions are met. Common criteria include a maximum number of iterations or convergence thresholds.

8. **Validation and Evaluation:**
    - Cross-validation is used to assess the selected feature subset's generalization performance on unseen data. Evaluation metrics such as accuracy, precision, recall, F1-score, and ROC curves are used to evaluate the model's performance.
9. **Results Visualization:**
    - Visualizations such as bar charts, heatmaps, or ROC curves may be generated to illustrate the performance of different feature subsets and help in the interpretation of results.
10. **Model Deployment:**
    - Once the optimal feature subset is selected, it can be used to train a predictive model for breast cancer diagnosis. This model can then be deployed for clinical use.
11. **Interpretability and Visualization (Optional):**
    - Depending on the chosen meta-heuristic algorithm, it may be helpful to visualize the selected features or their importance in the final model. Feature importance plots or explanations can aid in clinical interpretation.
12. **Documentation and Reporting:**
    - Documenting the entire process, including algorithm choice, parameter settings, and results, is essential for transparency and reproducibility. A report summarizing the findings and the selected feature subset is typically generated.
13. **Further Analysis (Optional):**
    - Additional statistical or clinical analysis may be conducted on the selected features to validate their relevance and potential biological significance.

4.2 SAMPLE CODE :

FunctionHO :

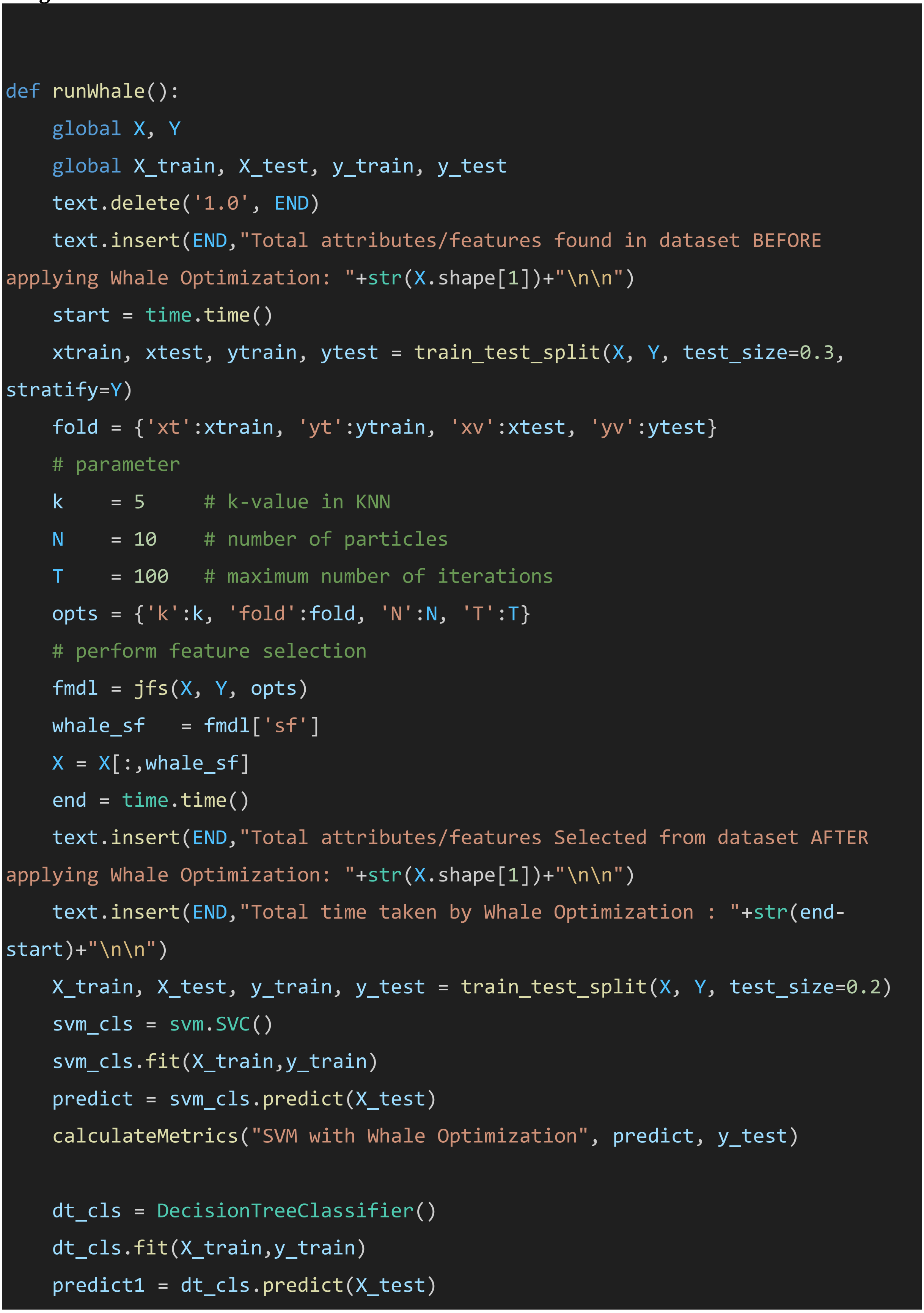


```python
import numpy as np
from sklearn.neighbors import KNeighborsClassifier


# error rate
def error_rate(xtrain, ytrain, x, opts):
    # parameters
    k     = opts['k']
    fold  = opts['fold']
    xt    = fold['xt']
    yt    = fold['yt']
    xv    = fold['xv']
    yv    = fold['yv']

    # Number of instances
    num_train = np.size(xt, 0)
    num_valid = np.size(xv, 0)
    # Define selected features
    xtrain  = xt[:, x == 1]
    ytrain  = yt.reshape(num_train)  # Solve bug
    xvalid  = xv[:, x == 1]
    yvalid  = yv.reshape(num_valid)  # Solve bug
    # Training
    mdl     = KNeighborsClassifier(n_neighbors = k)
    mdl.fit(xtrain, ytrain)
    # Prediction
    ypred   = mdl.predict(xvalid)
    acc     = np.sum(yvalid == ypred) / num_valid
    error   = 1 - acc
    return error



# Error rate & Feature size
def Fun(xtrain, ytrain, x, opts):
    # Parameters
    alpha     = 0.99
    beta      = 1 - alpha
    # Original feature size
    max_feat = len(x)
    # Number of selected features
    num_feat = np.sum(x == 1)
    # Solve if no feature selected
    if num_feat == 0:
        cost  = 1
    else:
        # Get error rate
        error = error_rate(xtrain, ytrain, x, opts)
        # Objective function
        cost  = alpha * error + beta * (num_feat / max_feat)

    return cost
```

Main.PY :

```
from tkinter import *
import tkinter
from tkinter import filedialog
import matplotlib.pyplot as plt
from tkinter.filedialog import askopenfilename
import numpy as np
from sklearn.metrics import accuracy_score
from sklearn.model_selection import train_test_split
from sklearn.metrics import confusion_matrix
```

```
import seaborn as sns
import pickle
from sklearn.metrics import precision_score
from sklearn.metrics import recall_score
from sklearn.metrics import f1_score
import os
from sklearn.preprocessing import LabelEncoder
from woa import jfs #importing woa whale feature selection class
import time
from sklearn import svm
from sklearn.tree import DecisionTreeClassifier
from sklearn.neighbors import KNeighborsClassifier
import pandas as pd
import pyswarms as ps #swarm package for PSO features selection algorithm
from SwarmPackagePy import testFunctions as tf
from sklearn import linear_model

main = tkinter.Tk()
main.title("META-HEURISTIC OPTIMIZATION")
main.geometry("1200x1200")

global X_train, X_test, y_train, y_test
global filename, dataset
global X, Y, XX
accuracy = []
precision = []
recall = []
fscore = []

lr_classifier = linear_model.LogisticRegression(max_iter=1000)

def uploadDataset():
    global filename, dataset
    text.delete('1.0', END)
    filename = filedialog.askopenfilename(initialdir="Dataset")
    text.insert(END,str(filename)+" Dataset Loaded\n\n")
    pathlabel.config(text=str(filename)+" Dataset Loaded\n\n")
    dataset = pd.read_csv("Dataset/WPBC.csv")
    text.insert(END,str(dataset.head()))
    label = dataset.groupby('diagnosis').size()
    label.plot(kind="bar")
    plt.title("Total number of Benign & Malignant Cases found in dataset")
    plt.show()


def preprocessDataset():
    global X, Y, dataset, XX
    global X_train, X_test, y_train, y_test
    text.delete('1.0', END)
    le = LabelEncoder()
    dataset.fillna(0, inplace = True)
    dataset['diagnosis'] =
pd.Series(le.fit_transform(dataset['diagnosis'].astype(str)))
    text.insert(END,str(dataset.head()))
    dataset = dataset.values
    Y = dataset[:,1:2].ravel()
    X = dataset[:,2:dataset.shape[1]-1]
    XX = X

def calculateMetrics(algorithm, predict, testY):
    p = precision_score(testY, predict,average='macro') * 100
    r = recall_score(testY, predict,average='macro') * 100
    f = f1_score(testY, predict,average='macro') * 100
    a = accuracy_score(testY,predict)*100
    text.insert(END,algorithm+' Accuracy  : '+str(a)+"\n")
    text.insert(END,algorithm+' Precision : '+str(p)+"\n")
    text.insert(END,algorithm+' Recall    : '+str(r)+"\n")
    text.insert(END,algorithm+' FMeasure  : '+str(f)+"\n\n")
    accuracy.append(a)
    precision.append(p)
    recall.append(r)
    fscore.append(f)
```



```python
def runWhale():
    global X, Y
    global X_train, X_test, y_train, y_test
    text.delete('1.0', END)
    text.insert(END,"Total attributes/features found in dataset BEFORE applying Whale Optimization: "+str(X.shape[1])+"\n\n")
    start = time.time()
    xtrain, xtest, ytrain, ytest = train_test_split(X, Y, test_size=0.3, stratify=Y)
    fold = {'xt':xtrain, 'yt':ytrain, 'xv':xtest, 'yv':ytest}
    # parameter
    k    = 5     # k-value in KNN
    N    = 10    # number of particles
    T    = 100   # maximum number of iterations
    opts = {'k':k, 'fold':fold, 'N':N, 'T':T}
    # perform feature selection
    fmdl = jfs(X, Y, opts)
    whale_sf    = fmdl['sf']
    X = X[:,whale_sf]
    end = time.time()
    text.insert(END,"Total attributes/features Selected from dataset AFTER applying Whale Optimization: "+str(X.shape[1])+"\n\n")
    text.insert(END,"Total time taken by Whale Optimization : "+str(end-start)+"\n\n")
    X_train, X_test, y_train, y_test = train_test_split(X, Y, test_size=0.2)
    svm_cls = svm.SVC()
    svm_cls.fit(X_train,y_train)
    predict = svm_cls.predict(X_test)
    calculateMetrics("SVM with Whale Optimization", predict, y_test)

    dt_cls = DecisionTreeClassifier()
    dt_cls.fit(X_train,y_train)
    predict1 = dt_cls.predict(X_test)
    calculateMetrics("Decision Tree with Whale Optimization", predict1,
y_test)

    knn_cls = KNeighborsClassifier(n_neighbors=2)
    knn_cls.fit(X_train,y_train)
    predict = knn_cls.predict(X_test)
    calculateMetrics("KNN with Whale Optimization", predict, y_test)
    LABELS = ['Benign', 'Malignant']
    conf_matrix = confusion_matrix(y_test, predict1)
    plt.figure(figsize =(6, 6))
    ax = sns.heatmap(conf_matrix, xticklabels = LABELS, yticklabels = LABELS,
annot = True, cmap="viridis" ,fmt ="g");
    ax.set_ylim([0,2])
    plt.title("Confusion matrix")
    plt.ylabel('True class')
    plt.xlabel('Predicted class')
    plt.show()


#PSO function to calculate importance of each features
def f_per_particle(m, alpha):
    global X
    global Y
    global lr_classifier
    total_features = 30
    if np.count_nonzero(m) == 0:
        X_subset = X
    else:
        X_subset = X[:,m==1]
    lr_classifier.fit(X_subset, Y)
    P = (lr_classifier.predict(X_subset) == Y).mean()
    j = (alpha * (1.0 - P) + (1.0 - alpha) * (1 - (X_subset.shape[1] /
total_features)))
    return j

def fun(x, alpha=0.88):
    n_particles = x.shape[0]
    j = [f_per_particle(x[i], alpha) for i in range(n_particles)]
    return np.array(j)


def runPSO():
    global X, Y, XX
    X = XX
    text.insert(END,"Total attributes/features found in dataset BEFORE applying PSO: "+str(X.shape[1])+"\n\n")
    start = time.time()
    options = {'c1': 0.5, 'c2': 0.5, 'w':0.9, 'k': 5, 'p':2}
    dimensions = X.shape[1] # dimensions should be the number of features
    optimizer = ps.discrete.BinaryPSO(n_particles=5, dimensions=dimensions, options=options) #CREATING PSO OBJECTS
    cost, pos = optimizer.optimize(fun, iters=2)#OPTIMIZING FEATURES
    X_selected_features = X[:,pos==1]  # PSO WILL SELECT IMPORTANT FEATURES WHERE VALUE IS 1
    end = time.time()
    text.insert(END,"Total attributes/features Selected from dataset AFTER applying PSO: "+str(X_selected_features.shape[1])+"\n\n")
    text.insert(END,"Total time taken by PSO: "+str(end-start)+"\n\n")

    X_train, X_test, y_train, y_test = train_test_split(X_selected_features, Y, test_size=0.2)
    svm_cls = svm.SVC()
    svm_cls.fit(X_train,y_train)
    predict = svm_cls.predict(X_test)
    calculateMetrics("SVM with PSO", predict, y_test)

    dt_cls = DecisionTreeClassifier()
    dt_cls.fit(X_train,y_train)
    predict1 = dt_cls.predict(X_test)
    calculateMetrics("Decision Tree with PSO", predict1, y_test)

    knn_cls = KNeighborsClassifier(n_neighbors=2)
    knn_cls.fit(X_train,y_train)
    predict = knn_cls.predict(X_test)
    calculateMetrics("KNN with PSO", predict, y_test)

    LABELS = ['Benign', 'Malignant']
    conf_matrix = confusion_matrix(y_test, predict1)
    plt.figure(figsize =(6, 6))
    ax = sns.heatmap(conf_matrix, xticklabels = LABELS, yticklabels = LABELS,
annot = True, cmap="viridis" ,fmt ="g");
    ax.set_ylim([0,2])
    plt.title("Confusion matrix")
    plt.ylabel('True class')
    plt.xlabel('Predicted class')
    plt.show()

def graph():
    df = pd.DataFrame([['Whale SVM','Precision',precision[0]],['Whale
SVM','Recall',recall[0]],['Whale SVM','F1 Score',fscore[0]],['Whale
SVM','Accuracy',accuracy[0]],
                       ['Whale Decision
Tree','Precision',precision[1]],['Whale Decision
Tree','Recall',recall[1]],['Whale Decision Tree','F1 Score',fscore[1]],['Whale
Decision Tree','Accuracy',accuracy[1]],
                       ['Whale KNN','Precision',precision[2]],['Whale
KNN','Recall',recall[2]],['Whale KNN','F1 Score',fscore[2]],['Whale
KNN','Accuracy',accuracy[2]],
                       ['PSO SVM','Precision',precision[3]],['PSO
SVM','Recall',recall[3]],['PSO SVM','F1 Score',fscore[3]],['PSO
SVM','Accuracy',accuracy[3]],
                       ['PSO Decision Tree','Precision',precision[4]],['PSO
Decision Tree','Recall',recall[4]],['PSO Decision Tree','F1
Score',fscore[4]],['PSO Decision Tree','Accuracy',accuracy[4]],
                       ['PSO KNN','Precision',precision[5]],['PSO
KNN','Recall',recall[5]],['PSO KNN','F1 Score',fscore[5]],['PSO
KNN','Accuracy',accuracy[5]],

                      ],columns=['Parameters','Algorithms','Value'])
    df.pivot("Parameters", "Algorithms", "Value").plot(kind='bar')
    plt.title("Whale & PSO Accuracy, Precision, Recall & FScore Graph")
    plt.show()



def close():
    main.destroy()

font = ('times', 14, 'bold')
title = Label(main, text='META-HEURISTIC OPTIMIZATION ALGORITHMS BASED FEATURE
SELECTION FOR CLINICAL BREAST CANCER DIAGNOSIS')
title.config(bg='DarkGoldenrod1', fg='black')
title.config(font=font)
title.config(height=3, width=120)
title.place(x=5,y=5)

font1 = ('times', 13, 'bold')
uploadButton = Button(main, text="Upload WPBC Dataset", command=uploadDataset)
uploadButton.place(x=50,y=100)
uploadButton.config(font=font1)

pathlabel = Label(main)
pathlabel.config(bg='brown', fg='white')
pathlabel.config(font=font1)
pathlabel.place(x=560,y=100)

preprocessButton = Button(main, text="Preprocess Dataset",
command=preprocessDataset)
preprocessButton.place(x=50,y=150)
preprocessButton.config(font=font1)

whaleButton = Button(main, text="Run SVM, Decision Tree & KNN with Whale
Optimization", command=runWhale)
whaleButton.place(x=50,y=200)
whaleButton.config(font=font1)
```

```
psoButton = Button(main, text="Run SVM, Decision Tree & KNN with PSO",
command=runPSO)
psoButton.place(x=50,y=250)
psoButton.config(font=font1)

graphButton = Button(main, text="Comparison Graph", command=graph)
graphButton.place(x=50,y=300)
graphButton.config(font=font1)

exitButton = Button(main, text="Exit", command=close)
exitButton.place(x=50,y=350)
exitButton.config(font=font1)


font1 = ('times', 12, 'bold')
text=Text(main,height=25,width=75)
scroll=Scrollbar(text)
text.configure(yscrollcommand=scroll.set)
text.place(x=500,y=150)
text.config(font=font1)


main.config(bg='LightSteelBlue1')
main.mainloop()
```

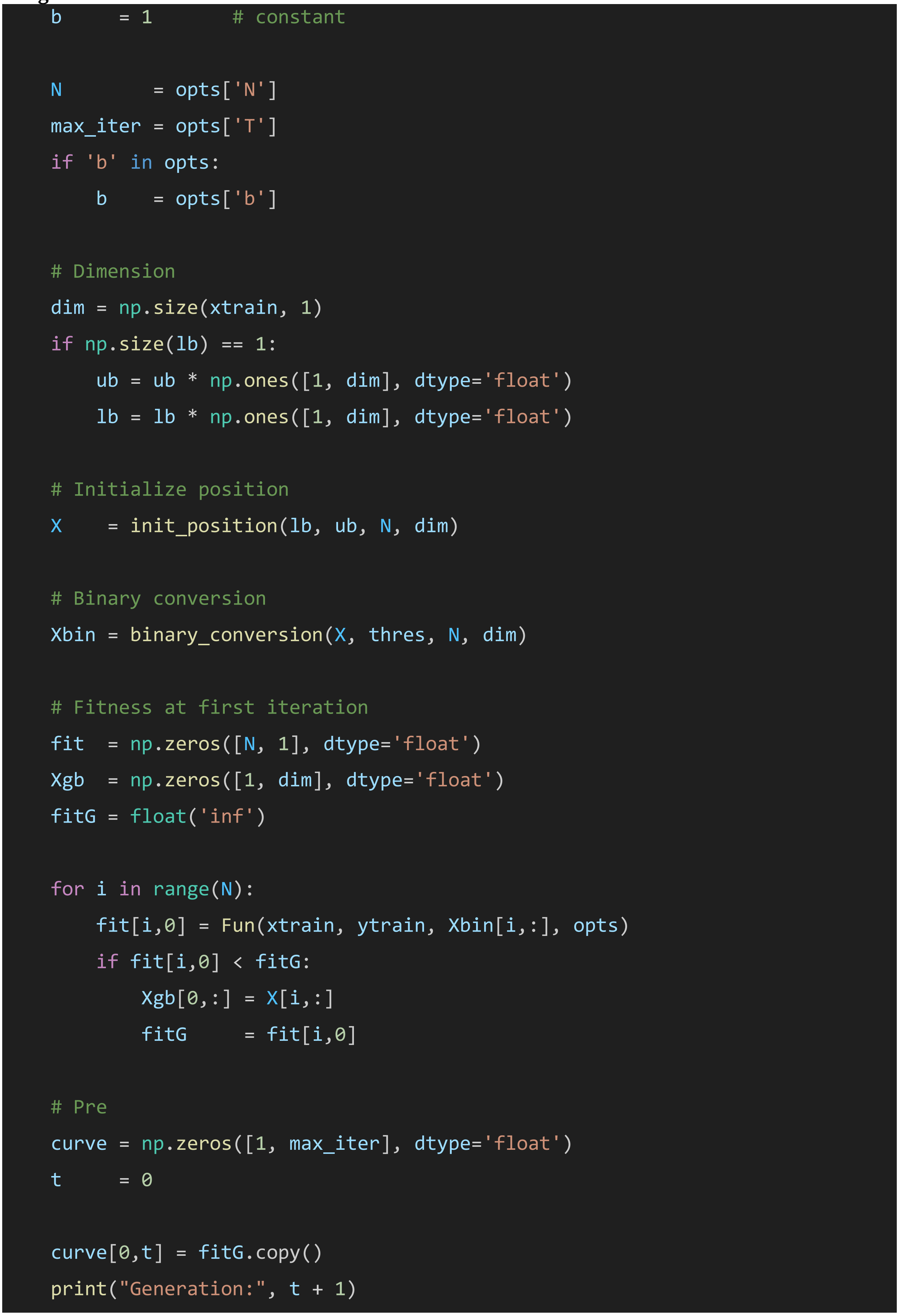
Test.PY :

```
import pandas as pd
import numpy as np
from sklearn.preprocessing import LabelEncoder
from woa import jfs
from sklearn.model_selection import train_test_split

le = LabelEncoder()
```

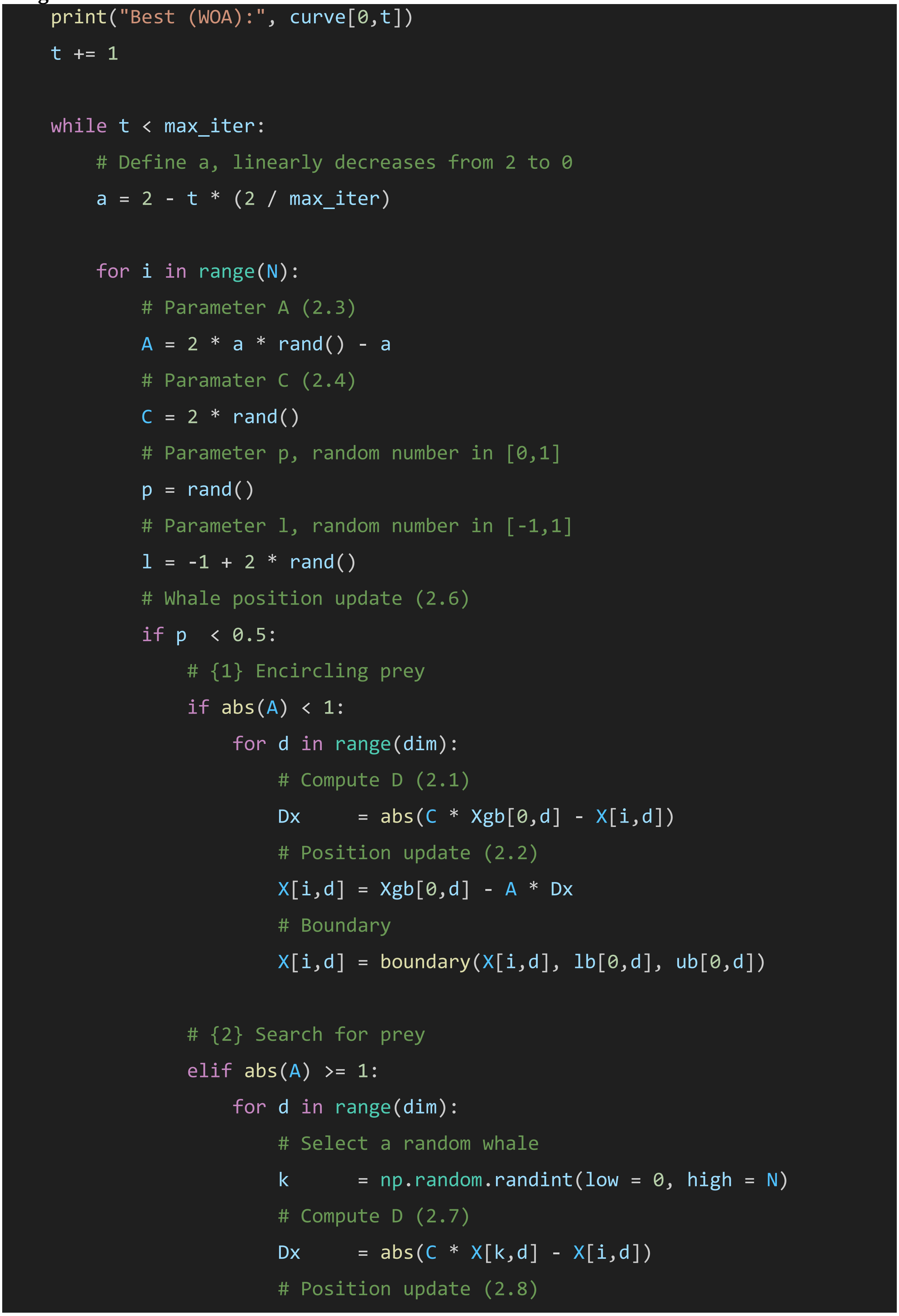
```
dataset = pd.read_csv("Dataset/WPBC.csv")
dataset.fillna(0, inplace = True)
dataset['diagnosis'] =
pd.Series(le.fit_transform(dataset['diagnosis'].astype(str)))
dataset = dataset.values

Y = dataset[:,1:2].ravel()
X = dataset[:,2:dataset.shape[1]-1]
print(X.shape)
xtrain, xtest, ytrain, ytest = train_test_split(X, Y, test_size=0.3,
stratify=Y)
fold = {'xt':xtrain, 'yt':ytrain, 'xv':xtest, 'yv':ytest}

# parameter
k    = 5     # k-value in KNN
N    = 10    # number of particles
T    = 100   # maximum number of iterations
opts = {'k':k, 'fold':fold, 'N':N, 'T':T}

# perform feature selection
fmdl = jfs(X, Y, opts)
sf   = fmdl['sf']

X = X[:,sf]

print(X.shape)
```

woa.PY :

```
#[2016]-"The whale optimization algorithm"]

import numpy as np
from numpy.random import rand
from functionHO import Fun


def init_position(lb, ub, N, dim):
    X = np.zeros([N, dim], dtype='float')
    for i in range(N):
        for d in range(dim):
            X[i,d] = lb[0,d] + (ub[0,d] - lb[0,d]) * rand()

    return X


def binary_conversion(X, thres, N, dim):
    Xbin = np.zeros([N, dim], dtype='int')
    for i in range(N):
        for d in range(dim):
            if X[i,d] > thres:
                Xbin[i,d] = 1
            else:
                Xbin[i,d] = 0

    return Xbin


def boundary(x, lb, ub):
    if x < lb:
        x = lb
    if x > ub:
        x = ub

    return x


def jfs(xtrain, ytrain, opts):
    # Parameters
    ub    = 1
    lb    = 0
    thres = 0.5
```



```
    b     = 1       # constant

    N        = opts['N']
    max_iter = opts['T']
    if 'b' in opts:
        b    = opts['b']

    # Dimension
    dim = np.size(xtrain, 1)
    if np.size(lb) == 1:
        ub = ub * np.ones([1, dim], dtype='float')
        lb = lb * np.ones([1, dim], dtype='float')

    # Initialize position
    X    = init_position(lb, ub, N, dim)

    # Binary conversion
    Xbin = binary_conversion(X, thres, N, dim)

    # Fitness at first iteration
    fit  = np.zeros([N, 1], dtype='float')
    Xgb  = np.zeros([1, dim], dtype='float')
    fitG = float('inf')

    for i in range(N):
        fit[i,0] = Fun(xtrain, ytrain, Xbin[i,:], opts)
        if fit[i,0] < fitG:
            Xgb[0,:] = X[i,:]
            fitG     = fit[i,0]

    # Pre
    curve = np.zeros([1, max_iter], dtype='float')
    t     = 0

    curve[0,t] = fitG.copy()
    print("Generation:", t + 1)
```



```
    print("Best (WOA):", curve[0,t])
    t += 1

    while t < max_iter:
        # Define a, linearly decreases from 2 to 0
        a = 2 - t * (2 / max_iter)

        for i in range(N):
            # Parameter A (2.3)
            A = 2 * a * rand() - a
            # Paramater C (2.4)
            C = 2 * rand()
            # Parameter p, random number in [0,1]
            p = rand()
            # Parameter l, random number in [-1,1]
            l = -1 + 2 * rand()
            # Whale position update (2.6)
            if p  < 0.5:
                # {1} Encircling prey
                if abs(A) < 1:
                    for d in range(dim):
                        # Compute D (2.1)
                        Dx     = abs(C * Xgb[0,d] - X[i,d])
                        # Position update (2.2)
                        X[i,d] = Xgb[0,d] - A * Dx
                        # Boundary
                        X[i,d] = boundary(X[i,d], lb[0,d], ub[0,d])

                # {2} Search for prey
                elif abs(A) >= 1:
                    for d in range(dim):
                        # Select a random whale
                        k      = np.random.randint(low = 0, high = N)
                        # Compute D (2.7)
                        Dx     = abs(C * X[k,d] - X[i,d])
                        # Position update (2.8)
                        X[i,d] = X[k,d] - A * Dx
                        # Boundary
                        X[i,d] = boundary(X[i,d], lb[0,d], ub[0,d])

            # {3} Bubble-net attacking
            elif p >= 0.5:
                for d in range(dim):
                    # Distance of whale to prey
                    dist   = abs(Xgb[0,d] - X[i,d])
                    # Position update (2.5)
                    X[i,d] = dist * np.exp(b * l) * np.cos(2 * np.pi * l) +
Xgb[0,d]
                    # Boundary
                    X[i,d] = boundary(X[i,d], lb[0,d], ub[0,d])

        # Binary conversion
        Xbin = binary_conversion(X, thres, N, dim)

        # Fitness
        for i in range(N):
            fit[i,0] = Fun(xtrain, ytrain, Xbin[i,:], opts)
            if fit[i,0] < fitG:
                Xgb[0,:] = X[i,:]
                fitG     = fit[i,0]

        # Store result
        curve[0,t] = fitG.copy()
        print("Generation:", t + 1)
        print("Best (WOA):", curve[0,t])
        t += 1


    # Best feature subset
    Gbin       = binary_conversion(Xgb, thres, 1, dim)
    Gbin       = Gbin.reshape(dim)
    pos        = np.asarray(range(0, dim))
    sel_index  = pos[Gbin == 1]
    num_feat   = len(sel_index)
    # Create dictionary
    woa_data = {'sf': sel_index, 'c': curve, 'nf': num_feat}

    return woa_data
```

5.TESTING

5.1 SYSTEM TESTING

The purpose of testing is to discover errors. Testing is the process of trying to discover every conceivable fault or weakness in a work product. It provides a way to check the functionality of components, sub assemblies, assemblies and/or a finished product It is the process of exercising software with the intent of ensuring that the
Software system meets its requirements and user expectations and does not fail in an unacceptable manner. There are
types of test. Each test type addresses a specific testing requirement.

TYPES OF TESTS

Unit testing:

Unit testing involves the design of test cases that validate that the internal program logic is functioning properly, and that program inputs produce valid outputs. All decision branches and internal code flow should be validated. It is the testing of individual software units of the application .it is done after the completion of an individual unit before integration. This is a structural testing, that relies on knowledge of its construction and is invasive. Unit tests perform basic tests at component level and test a specific business process, application, and/or system configuration. Unit tests ensure that each unique path of a business process performs accurately to the documented specifications and contains clearly defined inputs and expected results.

Integration testing

Integration tests are designed to test integrated software components to determine if they actually run as one program. Testing is event driven and is more concerned with the basic outcome of screens or fields. Integration tests demonstrate that although the components were individually satisfaction, as shown by successfully unit testing, the combination of components is correct and consistent. Integration testing is specifically aimed at exposing the problems that arise from the combination of components.

Functional test

Functional tests provide systematic demonstrations that functions tested are available as specified by the business and technical requirements, system documentation, and user manuals.

Functional testing is centered on the following items:

Valid Input : identified classes of valid input must be accepted.

Invalid Input : identified classes of invalid input must be rejected.

Functions : identified functions must be exercised.

Output : identified classes of application outputs must be exercised.

Systems/Procedures: interfacing systems or procedures must be invoked.

Organization and preparation of functional tests is focused on requirements, key functions, or special test cases. In addition, systematic coverage pertaining to identify Business process flows; data fields, predefined processes, and successive processes must be considered for testing. Before functional testing is complete, additional tests are identified and the effective value of current tests is determined.

System Test

System testing ensures that the entire integrated software system meets requirements. It tests a configuration to ensure known and predictable results. An example of system testing is the configuration oriented system integration test. System testing is based on process descriptions and flows, emphasizing pre-driven process links and integration points.

White Box Testing

White Box Testing is a testing in which in which the software tester has knowledge of the inner workings, structure and language of the software, or at least its purpose. It is purpose. It is used to test areas that cannot be reached from a black box level.

Black Box Testing

Black Box Testing is testing the software without any knowledge of the inner workings, structure or language of the module being tested. Black box tests, as most other kinds of tests, must be written from a definitive source document, such as specification or requirements document, such as specification or requirements document. It is a testing in which the software under test is treated, as a black box .you cannot “see” into it. The test provides inputs and responds to outputs without considering how the software works.

Unit Testing:

Unit testing is usually conducted as part of a combined code and unit test phase of the software lifecycle, although it is not uncommon for coding and unit testing to be conducted as two distinct phases.

Test strategy and approach

Field testing will be performed manually and functional tests will be written in detail.

Test objectives

- All field entries must work properly.
- Pages must be activated from the identified link.
- The entry screen, messages and responses must not be delayed.

Features to be tested

- Verify that the entries are of the correct format
- No duplicate entries should be allowed
- All links should take the user to the correct page.

Integration Testing

Software integration testing is the incremental integration testing of two or more integrated software components on a single platform to produce failures caused by interface defects.

The task of the integration test is to check that components or software applications, e.g. components in a software system or – one step up – software applications at the company level – interact without error.

Test Results: All the test cases mentioned above passed successfully. No defects encountered.

Test Results: All the test cases mentioned above passed successfully. No defects encountered.

SYSTEM TESTING

TESTING METHODOLOGIES

The following are the Testing Methodologies:

- **Unit Testing.**
- **Integration Testing.**
- **User Acceptance Testing.**
- **Output Testing.**
- **Validation Testing.**

Unit Testing

Unit testing focuses verification effort on the smallest unit of Software design that is the module. Unit testing exercises specific paths in a module's control structure to ensure complete coverage and maximum error detection. This test focuses on each module individually, ensuring that it functions properly as a unit. Hence, the naming is Unit Testing.

During this testing, each module is tested individually and the module interfaces are verified for the consistency with design specification. All important processing path are tested for the expected results. All error handling paths are also tested.

Integration Testing

Integration testing addresses the issues associated with the dual problems of verification and program construction. After the software has been integrated a set of high order tests are conducted. The main objective in this testing process is to take unit tested modules and builds a program structure that has been dictated by design.

The following are the types of Integration Testing:

1.Top Down Integration

This method is an incremental approach to the construction of program structure. Modules are integrated by moving downward through the control hierarchy, beginning with the main program module. The module subordinates to the main program module are incorporated into the structure in either a depth first or breadth first manner.

2. Bottom-up Integration

This method begins the construction and testing with the modules at the lowest level in the program structure. Since the modules are integrated from the bottom up, processing required for modules subordinate to a given level is always available and the need for stubs is eliminated. The bottom up integration strategy may be implemented with the following steps:

- The low-level modules are combined into clusters into clusters that perform a specific Software sub-function.
- A driver (i.e.) the control program for testing is written to coordinate test case input and output.
- The cluster is tested.
- Drivers are removed and clusters are combined moving upward in the program
  Structure

User Acceptance Testing

User Acceptance of a system is the key factor for the success of any system. The system under consideration is tested for user acceptance by constantly keeping in touch with the prospective system users at the time of developing and making changes wherever required. The system developed provides a friendly user interface that can easily be understood even by a person who is new to the system.

Output Testing

After performing the validation testing, the next step is output testing of the proposed system, since no system could be useful if it does not produce the required output in the specified format. Asking the users about the format required by them tests the outputs generated or displayed by the system under consideration. Hence the output format is considered in 2 ways – one is on screen and another in printed format.

Validation Checking

Validation checks are performed on the following fields.

Text Field:

The text field can contain only the number of characters lesser than or equal to its size. The text fields are alphanumeric in some tables and alphabetic in other tables. Incorrect entry always flashes and error message.

Numeric Field:

The numeric field can contain only numbers from 0 to 9. An entry of any character flashes an error messages. The individual modules are checked for accuracy and what it has to perform. Each module is subjected to test run along with sample data. The individually tested modules are integrated into a single system. Testing involves executing the real data information is used in the program the existence of any program defect is inferred from the output. The testing should be planned so that all the requirements are individually tested.

A successful test is one that gives out the defects for the inappropriate data and produces and output revealing the errors in the system.

Preparation of Test Data

Taking various kinds of test data does the above testing. Preparation of test data plays a vital role in the system testing. After preparing the test data the system under study is tested using that test data. While testing the system by using test data errors are

again uncovered and corrected by using above testing steps and corrections are also noted for future use.

Using Live Test Data:

Live test data are those that are actually extracted from organization files. After a system is partially constructed, programmers or analysts often ask users to key in a set of data from their normal activities. Then, the systems person uses this data as a way to partially test the system. In other instances, programmers or analysts extract a set of live data from the files and have them entered themselves.

It is difficult to obtain live data in sufficient amounts to conduct extensive testing. And, although it is realistic data that will show how the system will perform for the typical processing requirement, assuming that the live data entered are in fact typical, such data generally will not test all combinations or formats that can enter the system. This bias toward typical values then does not provide a true systems test and in fact ignores the cases most likely to cause system failure.

Using Artificial Test Data:

Artificial test data are created solely for test purposes, since they can be generated to test all combinations of formats and values. In other words, the artificial data, which can quickly be prepared by a data generating utility program in the information systems department, make possible the testing of all login and control paths through the program.

The most effective test programs use artificial test data generated by persons other than those who wrote the programs. Often, an independent team of testers formulates a testing plan, using the systems specifications.

The package "Virtual Private Network" has satisfied all the requirements specified as per software requirement specification and was accepted.

Testing Strategy

A strategy for system testing integrates system test cases and design techniques into a well planned series of steps that results in the successful construction of software. The testing strategy must co-operate test planning, test case design, test execution, and the resultant data collection and evaluation .A strategy for software testing must accommodate low-level tests that are necessary to verify that a small source code segment has been correctly implemented as well as high level tests that validate major system functions against user requirements.

Software testing is a critical element of software quality assurance and represents the ultimate review of specification design and coding. Testing represents an interesting anomaly for the software. Thus, a series of testing are performed for the proposed system before the system is ready for user acceptance testing.

System Testing

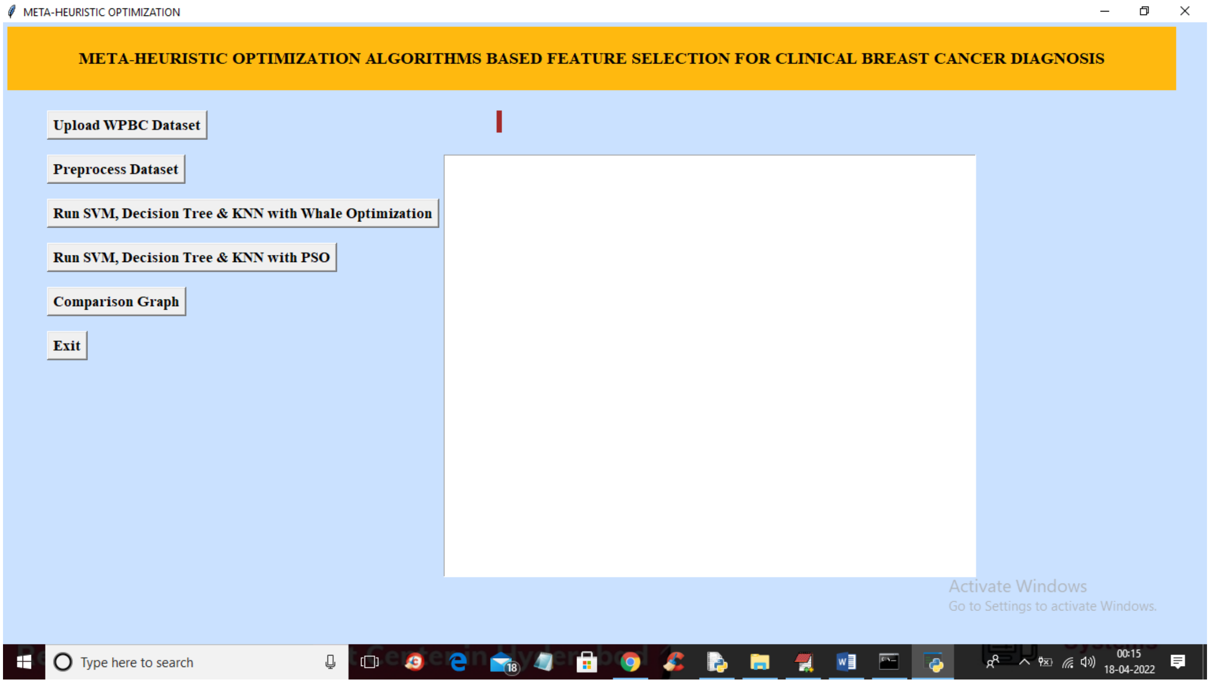
Software once validated must be combined with other system elements (e.g. Hardware, people, database). System testing verifies that all the elements are proper and that overall system function performance is achieved. It also tests to find discrepancies between the system and its original objective, current specifications and system documentation.

.

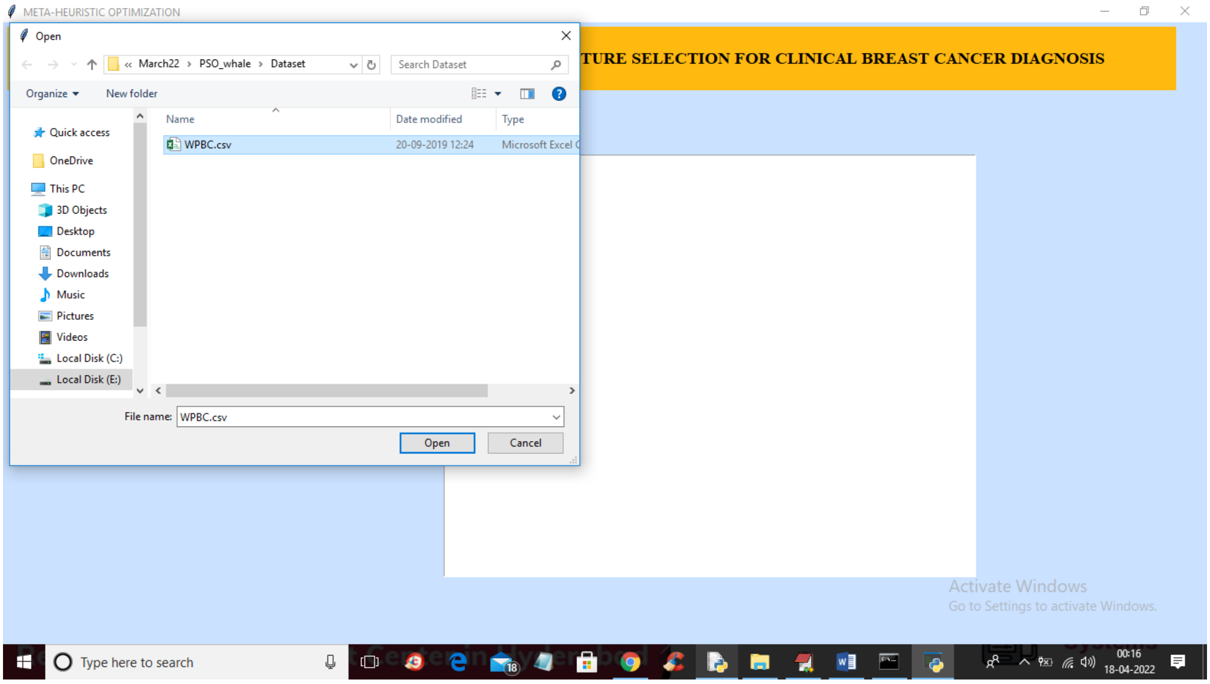
In Due Course, latest technology advancements will be taken into consideration. As part of technical build-up many components of the networking system will be generic in nature so that future projects can either use or interact with this. The future holds a lot to offer to the development and refinement of this project.

6.Result

To run project double click on ‘run.bat’ file to get below screen


META-HEURISTIC OPTIMIZATION
META-HEURISTIC OPTIMIZATION ALGORITHMS BASED FEATURE SELECTION FOR CLINICAL BREAST CANCER DIAGNOSIS
Upload WPBC Dataset
Preprocess Dataset
Run SVM, Decision Tree & KNN with Whale Optimization
Run SVM, Decision Tree & KNN with PSO
Comparison Graph
Exit
Activate Windows
Go to Settings to activate Windows.
Type here to search
00:15
18-04-2022

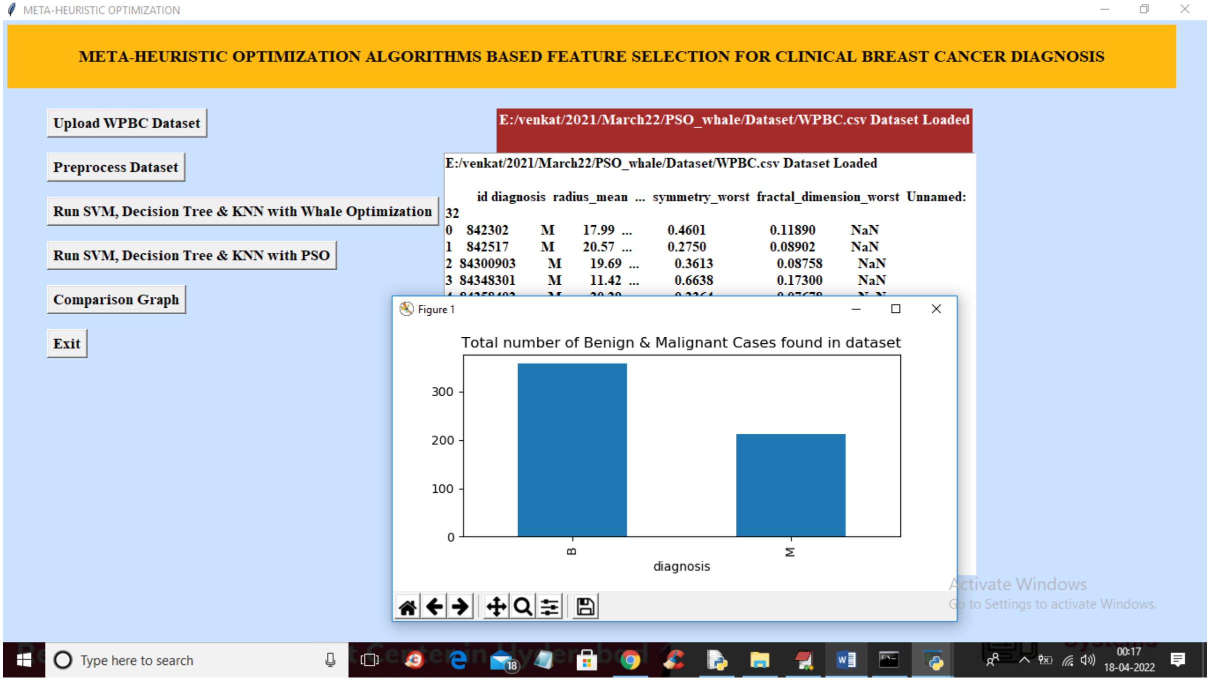
In above screen click on ‘Upload WPBC Dataset’ button to upload dataset and to get below screen



Name	Date modified	Type
WPBC.csv	20-09-2019 12:24	Microsoft Excel C

File name: WPBC.csv

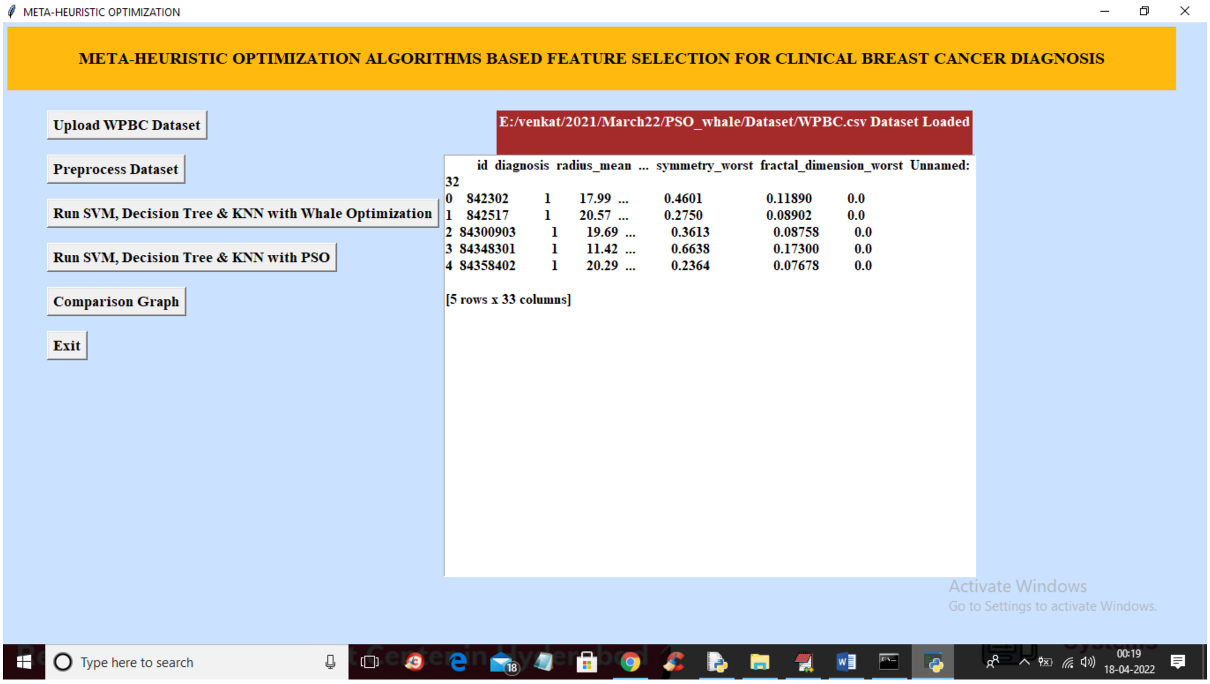
In above screen selecting and uploading dataset folder and then click on ‘Open’ button to load dataset and to get below screen


META-HEURISTIC OPTIMIZATION
META-HEURISTIC OPTIMIZATION ALGORITHMS BASED FEATURE SELECTION FOR CLINICAL BREAST CANCER DIAGNOSIS
Upload WPBC Dataset
Preprocess Dataset
Run SVM, Decision Tree & KNN with Whale Optimization
Run SVM, Decision Tree & KNN with PSO
Comparison Graph
Exit
Figure 1
Total number of Benign & Malignant Cases found in dataset
300
200
100
0
B
M
diagnosis
Activate Windows
Go to Settings to activate Windows.
Type here to search
00:17
18-04-2022

E:/venkat/2021/March22/PSO_whale/Dataset/WPBC.csv Dataset Loaded

```
E:/venkat/2021/March22/PSO_whale/Dataset/WPBC.csv Dataset Loaded

        id diagnosis  radius_mean  ...  symmetry_worst  fractal_dimension_worst  Unnamed:
32
0   842302         M        17.99  ...          0.4601                  0.11890        NaN
1   842517         M        20.57  ...          0.2750                  0.08902        NaN
2  84300903        M        19.69  ...          0.3613                  0.08758        NaN
3  84348301        M        11.42  ...          0.6638                  0.17300        NaN
```

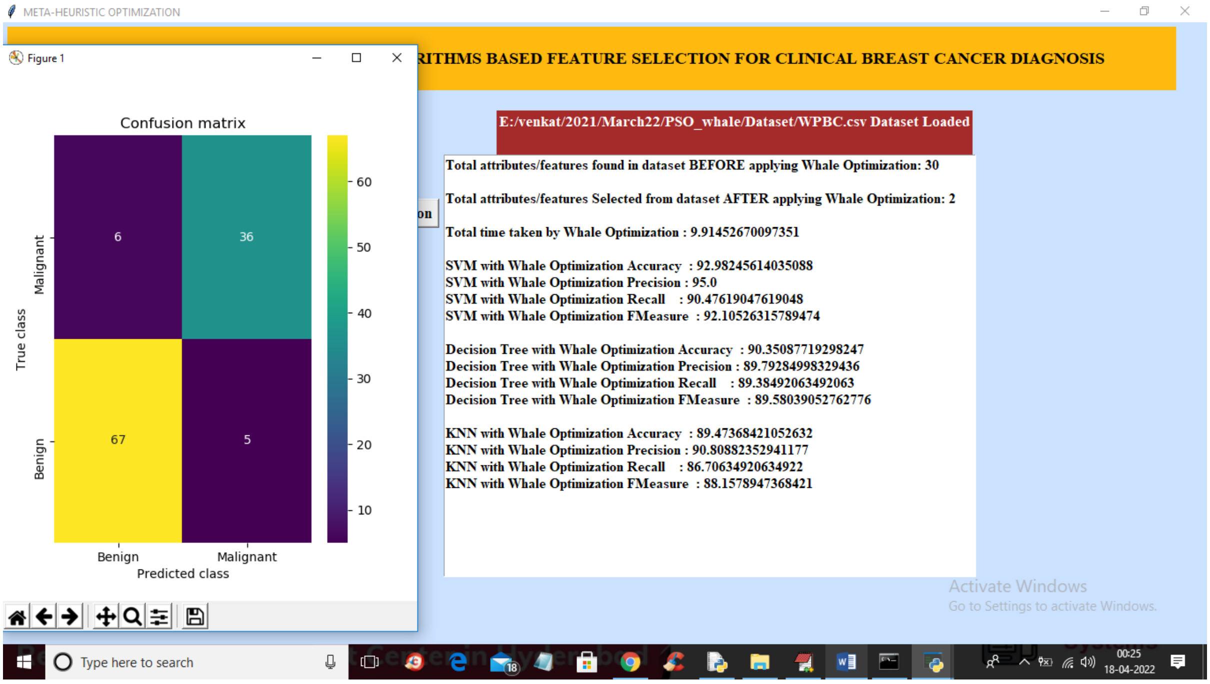
In above screen we can see dataset loaded and in dataset contains non-numeric values and missing NAN values and in above screen we can see graph showing number of benign and malignant cases found in dataset and now close above graph and then click on ‘Preprocess Dataset’ button to replace missing values and convert non-numeric data to numeric data and get below output


META-HEURISTIC OPTIMIZATION
META-HEURISTIC OPTIMIZATION ALGORITHMS BASED FEATURE SELECTION FOR CLINICAL BREAST CANCER DIAGNOSIS
Upload WPBC Dataset
Preprocess Dataset
Run SVM, Decision Tree & KNN with Whale Optimization
Run SVM, Decision Tree & KNN with PSO
Comparison Graph
Exit
Activate Windows
Go to Settings to activate Windows.
Type here to search
00:19
18-04-2022

E:/venkat/2021/March22/PSO_whale/Dataset/WPBC.csv Dataset Loaded

```
        id  diagnosis  radius_mean  ...  symmetry_worst  fractal_dimension_worst  Unnamed:
32
0   842302          1        17.99  ...          0.4601                  0.11890        0.0
1   842517          1        20.57  ...          0.2750                  0.08902        0.0
2  84300903         1        19.69  ...          0.3613                  0.08758        0.0
3  84348301         1        11.42  ...          0.6638                  0.17300        0.0
4  84358402         1        20.29  ...          0.2364                  0.07678        0.0

[5 rows x 33 columns]
```

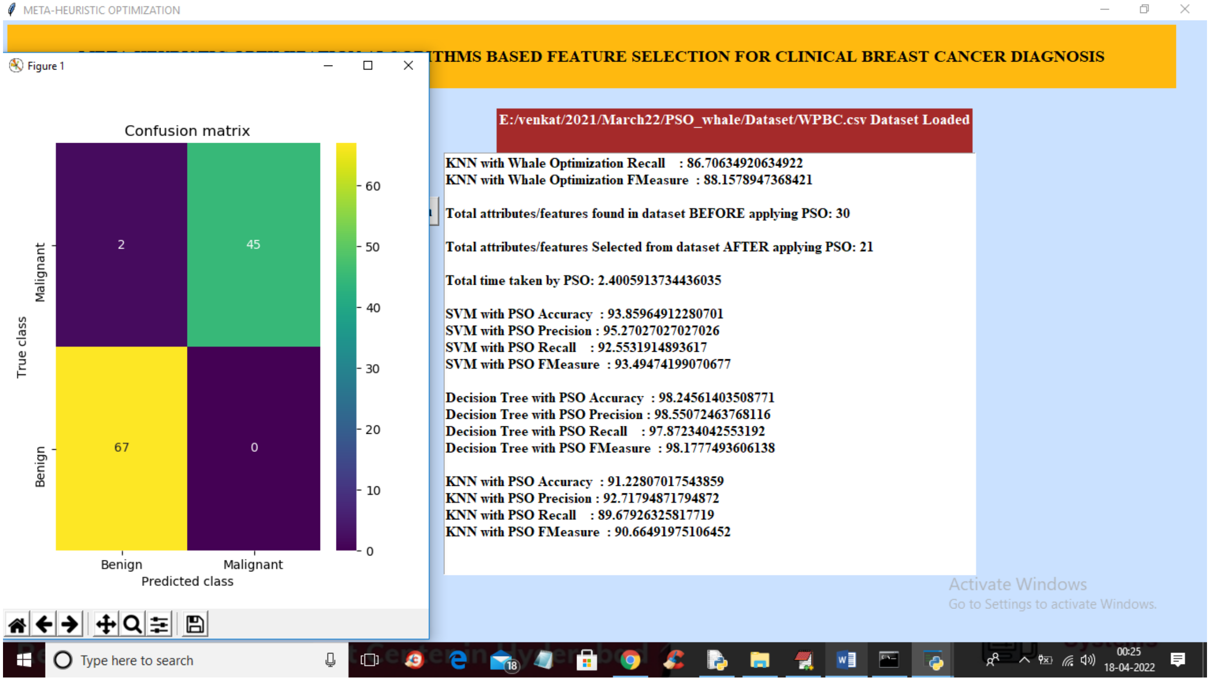
In above screen we can see all values are converted to numeric and now click on ‘Run SVM, Decision Tree & KNN with Whale Optimization’ button to apply whale optimization and train all ML algorithms to get below output



E:/venkat/2021/March22/PSO_whale/Dataset/WPBC.csv Dataset Loaded

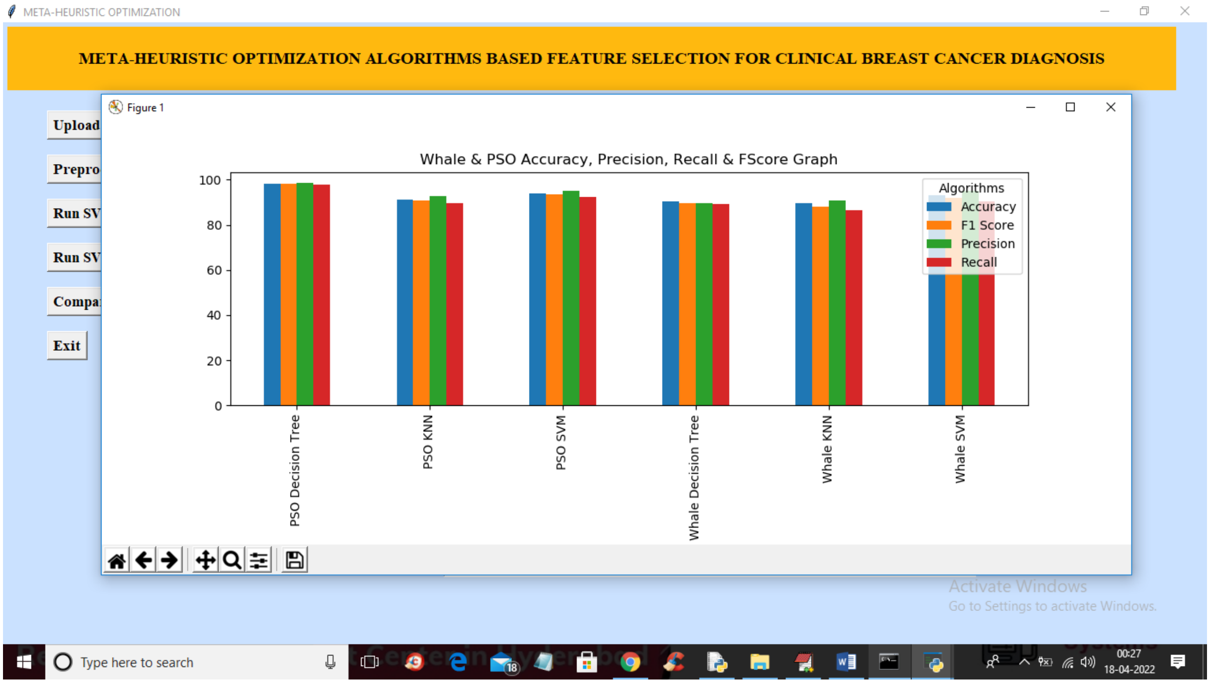
```
Total attributes/features found in dataset BEFORE applying Whale Optimization: 30

Total attributes/features Selected from dataset AFTER applying Whale Optimization: 2

Total time taken by Whale Optimization : 9.91452670097351

SVM with Whale Optimization Accuracy  : 92.98245614035088
SVM with Whale Optimization Precision : 95.0
SVM with Whale Optimization Recall    : 90.47619047619048
SVM with Whale Optimization FMeasure  : 92.10526315789474

Decision Tree with Whale Optimization Accuracy  : 90.35087719298247
Decision Tree with Whale Optimization Precision : 89.79284998329436
Decision Tree with Whale Optimization Recall    : 89.38492063492063
Decision Tree with Whale Optimization FMeasure  : 89.58039052762776

KNN with Whale Optimization Accuracy  : 89.47368421052632
KNN with Whale Optimization Precision : 90.80882352941177
KNN with Whale Optimization Recall    : 86.70634920634922
KNN with Whale Optimization FMeasure  : 88.1578947368421
```

In above screen in first two lines we can see dataset contains 30 attributes and after applying whale we got 2 important attributes and then we can see accuracy of each algorithm on selected features and we can see execution time also and in above screen we can see prediction confusion matrix graph where application predict 67 records as benign correctly and only 6 records are incorrectly predicted and now close above graph and then click on ‘Run SVM, Decision Tree & KNN with PSO’ button to select features with PSO and train all algorithms to get below output



```
E:/venkat/2021/March22/PSO_whale/Dataset/WPBC.csv Dataset Loaded

KNN with Whale Optimization Recall    : 86.70634920634922
KNN with Whale Optimization FMeasure  : 88.1578947368421

Total attributes/features found in dataset BEFORE applying PSO: 30

Total attributes/features Selected from dataset AFTER applying PSO: 21

Total time taken by PSO: 2.4005913734436035

SVM with PSO Accuracy  : 93.85964912280701
SVM with PSO Precision : 95.27027027027026
SVM with PSO Recall    : 92.5531914893617
SVM with PSO FMeasure  : 93.49474199070677

Decision Tree with PSO Accuracy  : 98.24561403508771
Decision Tree with PSO Precision : 98.55072463768116
Decision Tree with PSO Recall    : 97.87234042553192
Decision Tree with PSO FMeasure  : 98.1777493606138

KNN with PSO Accuracy  : 91.22807017543859
KNN with PSO Precision : 92.71794871794872
KNN with PSO Recall    : 89.67926325817719
KNN with PSO FMeasure  : 90.66491975106452
```

In above screen we can see PSO selected 21 attributes out of 30 and we can see accuracy of each algorithm on selected features and in confusion matrix we can see with PSO 67 records are correctly predicted as Benign and only 2 records are incorrectly predicted and now close above graph and then click on 'Comparison Graph' button to get below graph


META-HEURISTIC OPTIMIZATION
META-HEURISTIC OPTIMIZATION ALGORITHMS BASED FEATURE SELECTION FOR CLINICAL BREAST CANCER DIAGNOSIS
Upload
Prepro
Run SV
Run SV
Compa
Exit
Figure 1
Whale & PSO Accuracy, Precision, Recall & FScore Graph
100
80
60
40
20
0
PSO Decision Tree
PSO KNN
PSO SVM
Whale Decision Tree
Whale KNN
Whale SVM
Algorithms
Accuracy
F1 Score
Precision
Recall
Activate Windows
Go to Settings to activate Windows.
Type here to search
00:27
18-04-2022

In above graph x-axis represents algorithm names and y-axis represents accuracy and other values and in above graph each different colour bar represents different metrics like accuracy, precision and etc. In above graph we can see PSO with decision tree give better performance.

5. CONCLUSION

We conclude metaheuristic algorithms that are developed from 2009 to the 2019 year and their binary variants, which have been applied to feature selection problem. A detailed description and mathematical model of feature selection problem are given that could help researchers to understand the problem properly. Moreover, the techniques of solving feature selection problems are presented. Additionally, metaheuristic algorithms are considered in solving the feature selection problem. Therefore, basic definition, importance and the classification of metaheuristic algorithms are given. The evolutionbased, swarm-based, physics-based category, human rlated algorithms have been developed and applied to feature selection problems.

However, metaheuristic algorithms have some following drawbacks:

- They suffer from slow convergence rate due to random generation movement.
- They explore the search space without knowing the search direction.
- They can trap into local optima, or they have some premature convergence.
- The values of the parameters used in the metaheuristic algorithms have to be adjusted, this may also lead to pre-mature convergence.

Besides, the limitation of the metaheuristic algorithms, the modified and enhanced version of the algorithms were developed which are successfully applied to the feature selection problems. Also, a categorization is presented based on the behaviour of algorithms; evolution-based, swarm-based, physics-related and human behaviour related algorithms. This paper benefits in such a way that a list of metaheuristic algorithms is presented based on their classification. It also benefits for the application point of view as it consists of a case study. The case study presents the eight benchmark datasets and the optimal feature subsets are found by implementing different metaheuristic algorithms.

Evolution and human-related category, but there are several algorithms have been designed in the swarm and physics-related algorithms. It implies that there is a scope to develop or propose new metaheuristic algorithms in these categories. This paper mainly focuses on solving the feature selection problem using binary variants of metaheuristic algorithms. Hence, extensive literature is presented in every class of metaheuristic algorithms. All binary variants of all reviewed algorithms regarding feature selection problems are pointed. In swarm-based category, all binary variants of Cuckoo search, Bat algorithm, Firefly algorithm, flower pollination algorithm, Krill herd algorithm, Grey wolf optimizer, Ant lion optimizer, Dragonfly algorithm, Whale optimization algorithm, Grasshopper optimization algorithm, Salp swarm algorithm are reviewed with the key factor of solving feature selection problem. Moreover, hybrid approaches are also reviewed in the process of solving the feature selection problem.

It can be concluded that there is some area(s) which are less explored, such as spam detection, theft detection and weather prediction. However, lots of research has been done on the well-known datasets of UCI repository and in medical diagnosis (cancer classification), intrusion detection systems, text classification, multimedia etc. Hence, researchers should pay great attention to explore this area with metaheuristic algorithms. Moreover, there are some algorithms in the literature for which binary variants are not developed yet such as PFA, CGS, TCO, ES, HSO, WSA, BMO, OptBees, TGSR, EVOA, VCS, EPC, GbSA, CSO, WEO, LCA, EMA, VPL. These algorithms benefit classification after developing their binary version. From the literature, it can be observed that the researcher has to face many challenges to obtain the best feature subset of the considered classification problem. A good choice of classifier has a significant impact of the quality of obtained solution such KNN classifier is the most used classifier in getting the best subset with well-known datasets of UCI repository. After that, SVM classifier used to classify in different applications such as medical diagnosis, pattern recognition, image analysis etc. There are some other classifiers which are less used in terms of classification. Hence, this another gap to use different classifiers in classification problem and compared with most used ones. Finally, researchers will get the benefit of this study as they could find all the key factors in solving the feature selection problem using metaheuristic algorithms under one roof.

REFERENCES

[1] B. Xue, M. Zhang, W. N. Browne, and X. Yao, ‘‘A survey on evolutionary computation approaches to feature selection,’’ IEEE Trans. Evol. Comput., vol. 20, no. 4, pp. 606–626, Aug. 2016.

[2] P. Y. Lee, W. P. Loh, and J. F. Chin, ‘‘Feature selection in multimedia: The state-of-the-art review,’’ Image Vis. Comput., vol. 67, pp. 29–42, Nov. 2017.

[3] B. Remeseiro and V. Bolon-Canedo, ‘‘A review of feature selection methods in medical applications,’’ Comput. Biol. Med., vol. 112, Sep. 2019, Art. no. 103375.

[4] M. Sharma and P. Kaur, ‘‘A comprehensive analysis of nature-inspired meta-heuristic techniques for feature selection problem,’’ Arch. Comput. Methods Eng., pp. 1–25, Feb. 2020, doi: 10.1007/s11831-020-09412-6.

[5] M. Z. Asghar, A. Khan, S. Ahmad, and F. M. Kundi, ‘‘A review of feature extraction in sentiment analysis,’’ J. Basic Appl. Sci. Res., vol. 4, no. 3, pp. 181–186, 2014.

[6] Y. Saeys, I. Inza, and P. Larrañaga, ‘‘A review of feature selection techniques in bioinformatics,’’ Bioinformatics, vol. 23, no. 19, pp. 2507–2517, Oct. 2007.

[7] V. Bolón-Canedo and A. Alonso-Betanzos, ‘‘Ensembles for feature selection: A review and future trends,’’ Inf. Fusion, vol. 52, pp. 1–12, Dec. 2019.

[8] H. Liu and L. Yu, ‘‘Toward integrating feature selection algorithms for classification and clustering,’’ IEEE Trans. Knowl. Data Eng., vol. 17, no. 4, pp. 491–502, Apr. 2005.

[9] S. Ahmed, M. Zhang, and L. Peng, ‘‘Enhanced feature selection for biomarker discovery in LC-MS data using GP,’’ in Proc. IEEE Congr. Evol. Comput., Jun. 2013, pp. 584–591.

[10] M. H. Aghdam, N. Ghasem-Aghaee, and M. E. Basiri, ‘‘Text feature selection using ant colony optimization,’’ Expert Syst. Appl., vol. 36, no. 3, pp. 6843–6853, Apr. 2009.

[11] A. Ghosh, A. Datta, and S. Ghosh, ‘‘Self-adaptive differential evolution for feature selection in hyperspectral image data,’’ Appl. Soft Comput., vol. 13, no. 4, pp. 1969–1977, Apr. 2013.

[12] M. Dash and H. Liu, ‘‘Feature selection for classification,’’ Intell. Data Anal., vol. 1, nos. 1–4, pp. 131–156, 1997.

[13] I. Guyon and A. Elisseeff, ‘‘An introduction to variable and feature selection,’’ J. Mach. Learn. Res., vol. 3, pp. 1157–1182, Mar. 2003.

[14] H. Liu, H. Motoda, R. Setiono, and Z. Zhao, ‘‘Feature selection: An ever evolving frontier in data mining,’’ in Feature Selection in Data Mining. Hyderabad, India, 2010, pp. 4–13.

[15] N. Hoque, D. K. Bhattacharyya, and J. K. Kalita, ‘‘MIFS-ND: A mutual information-based feature selection method,’’ Expert Syst. Appl., vol. 41, no. 14, pp. 6371–6385, Oct. 2014.

[16] Z. Xu, I. King, M. R.-T. Lyu, and R. Jin, ‘‘Discriminative semi-supervised feature selection via manifold regularization,’’ IEEE Trans. Neural Netw., vol. 21, no. 7, pp. 1033–1047, Jul. 2010.

[17] J. Tang, S. Alelyani, and H. Liu, ‘‘Feature selection for classification: A review,’’ in Data Classification: Algorithms and Applications. 2014, p. 37.

[18] A. Jović, K. Brkić, and N. Bogunović, ‘‘A review of feature selection methods with applications,’’ in Proc. 38th Int. Conv. Inf. Commun. Technol., Electron. Microelectron. (MIPRO), 2015, pp. 1200–1205.

[19] Z. Sun, G. Bebis, and R. Miller, ‘‘Object detection using feature subset selection,’’ Pattern Recognit., vol. 37, no. 11, pp. 2165–2176, Nov. 2004.

[20] A. K. Jain, R. P. W. Duin, and J. Mao, ‘‘Statistical pattern recognition: A review,’’ IEEE Trans. Pattern Anal. Mach. Intell., vol. 22, no. 1, pp. 4–37, Jan. 2000.

[21] H. Liu and H. Motoda, Feature Extraction, Construction Selection: A Data Mining Perspective, vol. 453. USA: Springer, 1998.