UNIVERSITY OF KWAZULU-NATAL™

INYUVESI YAKWAZULU-NATALI

University of KwaZulu-Natal School of Maths, Stats, and Computer Science Mini Project

Deep neural networks based model for Classifying Diabetes Mellitus using metaheuristic optimization algorithms for feature selection

Authors:

Nikita Calista Tirumol

University of KwaZulu-Natal

School of Maths, Statistics and Computer Science

217014607@stu.ukzn.ac.za

Shankar Bhawani Dayal

University of KwaZulu-Natal

School of Maths, Statistics and Computer Science

217027128@stu.ukzn.ac.za

Shaneer Luchman

University of KwaZulu-Natal

School of Maths, Statistics and Computer Science

216003502@stu.ukzn.ac.za

Abstract - Diabetes cases are escalating rapidly across the globe. Early diagnosis is necessary in managing the condition, preventing the risk of serious complications as well as the treatment needed to ensure the condition is managed is relatively expensive and hence the detecting diabetes is vital, and it needs to be accurate. The paper presents a deep neural network that uses feature selection techniques to select the best features that will result in the best accuracy regarding diabetes diagnosis. Feature selection is implemented using metaheuristics GA, PSO and a new hybrid PSO and GA method. After which the best features are fed into neural network and trained on it. Experimental results are performed on processed and unprocessed data. The results revealed that the PSO and GA hybrid had the best accuracy concerning processed data whilst GA had best accuracy with regards to unprocessed data.

Keywords – Diabetes mellitus, PIMA Indian diabetes dataset, Particle swarm optimization, Genetic Algorithm, PSO-GA hybrid, Deep neural network

1. Introduction

Diabetes or diabetes mellitus is a collection of metabolic diseases where the body is unable to metabolise glucose, thus the blood sugar levels are high for an extended period of time [1]. This disease destroys the ability to produce insulin in the patient's body or the body develops resistance to insulin and consequently the produced insulin cannot perform as intended. The role of insulin is to regulate blood sugar levels, after a person eats the carbohydrates from the food are broken down into glucose. Glucose then enters the bloodstream, after which the pancreas will produce insulin, allowing glucose to enter body cells providing energy.

In this paper we have decided to use metaheuristic optimization algorithms for feature selection. We aimed to optimize a neural network by only selecting the best features in the dataset for training. Feature Selection, also known as variable or attribute selection is the process of choosing the most significant features from a given dataset [2]. In many cases it is used to enhance the performance of machine learning models by not only improving accuracy but with less features to process, computational time is also improved [2]. Since the topic we were tasked with was the optimization of a neural network trained for predicting diabetes, we had to work with a medical dataset, which tend to possess many features for classification but also have a small number of patient records. The combination of the previously mentioned issues meant that some of the data contained in the dataset are noisy and irrelevant which will affect the performance of the neural network. We proposed the implementation of 3 metaheuristic algorithms (PSO, GA, hybrid PSO-GA) for selecting the most prominent features from the PIMA Indian dataset to remove these redundancies and improve performance.

1.1. Problem Description

Diabetes or diabetes mellitus is a chronic disease that affects a vast majority of the people in many parts of the world. If diabetes is left untreated, it can cause heart disease, stroke, kidney damage and nerve damage [3]. According to the International Diabetes Federation [4], in 2019 approximately 463 million adults, between the ages of 20 and 79 years, were living with diabetes, and by the year 2045 this will rise to 700 million adults living with diabetes. The International Diabetes Federation [4] also stated that in 2019 approximately 1 in 2 (232 million) people with diabetes were undiagnosed.

Machine learning can help people make a preliminary diagnosis in regard to whether they have diabetes or not, based upon their physical examination data, and it can serve as a reference for doctors [5, 6, 7]. Selecting important features and discarding the unnecessary features is an important problem for any

machine learning method, since unimportant features may cause the accuracy of the machine learning method to be impacted negatively.

This project aims improve on the study by S. I. Ayon, and M. M. Islam [8], by using metaheuristic algorithms for feature selection. In [8] they created a deep neural network to classify if a person has diabetes or not. For this project, we aim to minimize the number of features selected while maximizing the accuracy of the neural network.

2. Literature Review

This study [8] utilizes a deep neural network with 4 hidden layers and 1 input and output layer respectively, where the number of nodes for the hidden layers are 12, 14, 14 and 16 respectively. They trained their network using k-fold cross validation, they trained it with five-fold and 10-fold to compare the results. The claimed accuracy of the deep neural network using 10-fold cross validation is of 97.11%, upon trying to validate these results, this project created the model exactly as described and only achieved an accuracy of 75.40%. The deep neural network in [8] will serve as the base model for this project.

In 2013, Fayssal Beloufa and M.A. Chikh [9] proposed a diabetes prediction technique using a fuzzy classifier and an Artificial Bee Colony (ABC) metaheuristic. The technique improves the diversification and intensification of the search space using a blended crossover over operator on condition the current solution can't be revised. In [9] the technique was demonstrated using PIMA indian diabetes dataset and results showed improved performance.

In 2014, Sarab AlMuhaideb and Mohamed El Bachir Menai [10] proposed a hybrid technique called hybrid ant–bee colonies (HColonies), which is used for diagnosing medical conditions such as diabetes. The algorithm consists of two stages: ant colony stage (ACO) which is used to initialize the population and artificial bee colony stage (ABC) which is utilized for the metaheuristic. ABC is used to optimize the search by optimizing the decision lists that are created by the AntMiner+ algorithm, thereby improving performance. Experimental results show the technique is robust to variable parameters. The technique becomes inefficient with respect to multiclass classification but since diabetes prediction is a binary classification, this should not be a problem.

In 2012, Mohammad Reza Daliri [11] stated feature selection will assist in selecting the most relevant features to diagnose a particular medical condition, leading to a higher classification accuracy and therefore requiring minimal human intervention. The technique proposed utilized binary particle swarm optimization (PSO) for medical diagnosis and support vector machines to calculate the fitness for binary PSO. The technique has been implemented on several binary medical domain problems including diabetes prediction using PIMA indian diabetes dataset. The experimental results on the PIMA indian diabetes dataset were promising with the technique outperforming all techniques it was reviewed against and a relatively high prediction accuracy.

In 2017, Karamath Ateeq and Dr. Gopinath Ganapathy [12] proposed a hybrid algorithm referred to as MPSO-NN. This hybrid was created by modifying the PSO algorithm and integrating the modified PSO with a multi-layer perceptron network using back propagation learning. The datasets were preprocessed using Genetic-Relative Reduct Algorithm. Experimental results showed the technique had promising results with respect to accuracy.

In 2015, Hamid Reza Sahebi and Sara Ebrahimi [13] presented a fuzzy classifier to predict diabetes and a feature selection technique. A modified binary PSO technique is used to derive a set of fuzzy rules for prediction of diabetes disease. Methods such as classification rate, sensitivity and specificity values were used to evaluate performance. Experimental results showed the method achieved highest

accuracy when predicting amongst popular and newer techniques. The high classification rate was procured using a basic and interpretable fuzzy rule base.

In 2015, E Sreedevi and Prof. M Padmavathamma [14] proposed a threshold genetic algorithm to diagnose diabetes. The GA fitness function is calculated using Minkowski Distance Method and a selection operator of tournament selection is utilized. The distance method is used to procure optimized and accurate results. Experimental results were based on accuracy when using different distance fitness functions. The use of the Minkowski Distance Method outperformed the results of the other distance methods it was compared to.

In 2017 Kumar, S [2] proposed a feature selection technique combining Naive Bayes; a probabilistic approach and genetic algorithm to improve accuracy of diabetes diagnosis. The use of genetic algorithms operator's mutation and crossover encourage variation regarding the solutions and prevent getting trapped in local optima. The paper showed an increase accuracy of 1,732% increase with a lower computational time.

3. Choice of Technique

In this project we used a total of 3 techniques, 2 of which are metaheuristic algorithms and the last one is a hybrid of the metaheuristic algorithms. The metaheuristic algorithms that we implemented are particle swarm optimization (PSO) and genetic algorithm (GA) for feature selection. The hybrid algorithm that we implemented utilizes both PSO and GA for feature selection.

a) Genetic Algorithm (GA)

GA was mentioned as early on in 1975 by J. Holland in Adaptation in natural and artificial systems [15,16]. It is classed as one of the evolutionary-based heuristics, inspired by Charles Darwin's theory of natural selection [1]. GA has been used for feature extraction for a wide variety of problems [17, 18, 19, 20, 21], thus we decided to implement our own version of GA for feature selection. One of the main advantages of GA is that it can explore the search space in many directions simultaneously since each offspring or individual of the population act independently from each other, this feature is makes it ideal to parallelise the methods for implementation [22]. GA utilizes mutation which prevents the algorithm from converging, mutation does this by creating diversity in the population.

The main reasons for choosing GA for feature selection is:

- It is not problem specific, which allows us to adapt the algorithm to suit our needs.
- They are effective for problems with large search spaces due to crossover and mutation.
- They tend to have a good balance between intensification and diversification.
- As stated above, the parallelism of GA is a feature which we find attractive for this problem.

GA follows the principle of "survival of the fittest". The main components of the Genetic Algorithm are described below:

Chromosome – Each chromosome in the population represents a possible solution to the problem. A possible representation of a chromosome can be a "bit-string" or binary string containing '1' and '0'. These chromosomes have different genes(values) that can be used to describe and differentiate different chromosomes. An example would of this would be seen in implementations to solve the 0-1 knapsack problem [23, 24]. Each chromosome has a fitness function which would be problem dependant.

Initial population – A population would be created full of chromosomes based on the requirements for the chromosome. Each chromosome would then be ranked based on its fitness value. The initial population can be described as the set of randomly generated solutions and the population size varies according the problem.

Fitness Function – This will determine the fitness of a chromosome, it serves as a method of evaluating a chromosome and allowing chromosomes to be compared to each other.

Selection Method – The selection method of GA is very crucial, since it used to determine which chromosomes would be chosen for reproduction for the next generation. There are multiple selection methods to choose from, the most popular selection method are:

- Elitism- Choosing the best or worst chromosomes from the population based on their fitness value, this can be problem dependent.
- Tournament selection- This involves selecting random individuals from the population and choosing the best among them based on their fitness value. This is repeated until you have the desired number of chromosomes selected
- Roulette Wheel Selection- Roulette wheel selection is also known as Fitness proportionate selection, each chromosome is given a probability to be chosen based on their fitness value.

There are other selection methods, each one has its own advantage and disadvantage, and the one chosen is generally problem dependent. The option to combine selection method is also possible.

Crossover Operator – This process is responsible for the reproduction of the chosen chromosomes from the selection method. This method mimics the process of mating and reproduction in nature. For crossover, two chromosomes are selected (parents), and then a crossover point is chosen which can be randomised, and the two new chromosomes will be created (children) with genes from the parents. There is no guarantee that the offspring will be better than the parents, but this is good since it serves as diversification which is desired quality in metaheuristic algorithms. The chance for chromosomes to be selected for reproduction is based on the crossover rate.

Parents								Children							
Parent 1								Parent 1							
0	1	0	1	1	1	1	1	0	1	0	1	0	0	0	0
Parent 2								Parent 2							
1	0	1	0	0	0	0	0	1	0	1	0	1	1	1	1

Table 1: This is an example of crossover with the crossover point being in the middle of the chromosome

Mutation Operator- This method is responsible for altering chromosomes in the population, which results in diversity in the population. This is desirable since it lowers the chances of the GA of being trapped in a local optimum. Generally, mutation changes one gene in a chromosome, a random gene is selected and then is mutated/changed. The chance of a chromosome undergoing mutation is dependent on the mutation rate. The mutation rate is generally kept low otherwise it would result in the GA being a random search.

Chromosome before and after mutation

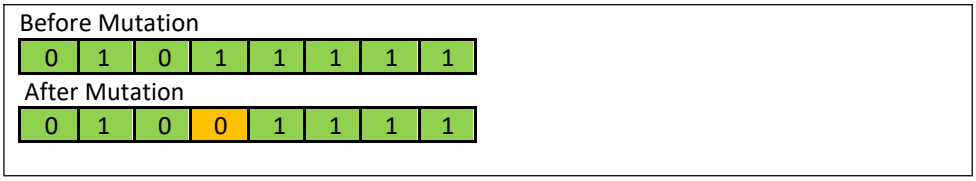


Table 2: Chromosome before and after mutation

Termination Criteria/Stopping condition – This determines when the GA should stop, the termination criteria to run until you reach the max generation, which will be user defined, or run for a set amount of time, or run until the algorithm reaches a set amount of convergence to a specific chromosome.

The general algorithm for GA can be summarised as follows:

1. **Randomly initialize population and set the parameters**
2. **Calculate fitness of each chromosome and rank them**
3. **UNTIL Termination Criteria met, REPEAT:**
    a. **Select parents from previous population using the selection method**
    b. **Apply crossover method and generate new population**
    c. **Apply mutation method on the new generation**
    d. **Calculate fitness for new population and rank them**
4. **Output best chromosome**

b) Particle Swarm Optimization (PSO)

PSO was introduced by J. Kennedy and R. Eberhart in 1995 [25]. PSO is an evolutionary algorithm [26] which was inspired by the flocking behaviour of fish and birds. PSO is different from GA since there is shared knowledge between the particles in PSO, which is the known as the global best particle/individual encountered in the swarm. PSO has also been used for feature selection [27, 28, 29, 30]. The contrast of PSO to GA is that PSO has a reputation of converging prematurely and getting trapped in a local minimum [31]. Since every particle in the swarm is changed in each new generation, PSO has a higher probability of finding the best solution compared to GA [31].

The PSO algorithm performs searching using a swarm of particles that get updated every iteration/generation [32]. The PSO algorithm attempts to find an optimal solution by moving each particle in the direction to the global best position in the swarm and its previously best position.

Initially, each particle is assigned a random generated position, along with a random initial velocity. The initial velocity has to be within a specified range being *Vmin* and *Vmax*. The fitness of every particle is then calculated, and the global best position is identified. In the first iteration/population, the personal best position of each particle is their current position, it gets updated in further iterations with changes to its velocity which changes its position.

The velocity of the particle *i* is updated as follows:

$$(+ 1) = (\times ( )) + _1 _1(\quad - ( )) + _2 _2(\quad - ( )) \tag{1}$$

The position of particle *i* is updated as follows:

$$(+1)= (+1)+ ( ) \quad (2)$$

Where:

- h
- h
- $_1$ h h ( )
- $_1$ h [0, 1]
- $_2$ h [0, 1]
- $_2$ h h ( )
- h h h h

- h h

- h

The velocity of the particle has to be in the range of [Vmin, Vmax].

The general algorithm of PSO is as follows [33]:

Step 1. Initialize swarm:

For each particle *i* = 1,2,3,……,n do

a. **Initialize the particle *i* with a random position and velocity**
b. **Set the pbest of *i* to initial position**
c. **Calculate the fitness of each particle**
d. **Select the particle with the best fitness function as gbest**

Step 2. Repeat until Termination criteria is met:

For each particle *i* = *1,2,3,*…….,n do

a. **Generate random numbers,** **and** $\in [,]$
b. **Update particle *i's* velocity. Refer to equation (1)**
c. **Update particles *i's* position. Refer to equation (2)**
d. **IF fitness of particle *i* at t+1 is better thanTHEN:**
    1. **Updateto the the new best position of particle *i***
    2. **IF fitness of particle *i* at new position is better than gbest THEN:**
        i. **Update gbest to the new position of particle *i***

e. $= +$

Step 3:

Output gbest, which would be the best encountered solution

c) PSO-GA

In this project we decided to create a hybrid metaheuristic, where we decided to tackle the shortcomings of the PSO algorithm with the advantages of GA. We decided to implement crossover and mutation on the fittest 30% of the population to lower the chances of the PSO algorithm from converging. PSO-GA hybrids have been created before [34, 35], and they have also been used for feature selection in other papers [36, 37]. The implementation of a PSO-GA hybrid is problem dependent and can be modelled in several ways.

4. Methodology

The deep neural network (DNN) that we have chosen to base our paper is from [8]. This DNN has four hidden layers, with 1 input layer and output layer, where the number of nodes/neurons for the hidden layers are 12, 14, 14 and 16 respectively. We have modelled the neural network just as described in [8].

For all our metaheuristic feature selection methods, we implemented the same fitness function. This is essentially since each of our implementations are working with the same solution representation, i.e. the GA chromosome and the PSO particle is represented as an array, and they both are trying to maximize the accuracy of the solution.

4.1. GA

We selected the GA for its high adaptability, and so we proposed an implementation of it for selecting the best features with a high correlation in the dataset, for training the neural network.

Chromosomal Representation – The chromosome is represented by a binary list(only contains 1's and 0's) with the length being the total number of features in the dataset, where each feature if selected is represented by 1 and if not, represented by 0. E.g. [0, 1, 0, 1, 1, 0, 1, 0] would represent 4 features being selected.

Fitness Function – To measure the optimality of a solution using a loss function which returned a value between [0,1]. The equation is as follows:

$$=(*)+(1-)* \quad (3)$$

Where:

- 01
- =(1- h h 5)
- =

Selection Operator – The selection technique chosen is Roulette Wheel Selection, because it granted a good balance between taking the better solutions and taking random solutions from the population. The steps of Roulette Wheel selection are described below:

1. Add all fitness values in the population to get **sum**

2. Generate a random number **rand** between 0 and **sum**

3. Starting at **rand**, iteratively add each fitness value (starting with the highest) until that new value is greater than **sum**. Whenever that iteration is complete, select the chromosome that was last added.

Crossover Operator – We initially select 2 individuals as parents using the Roulette wheel selection technique, then a random value between [0,1] is generated. If that value is less than the CR, crossover occurs. This is done by finding the middle point of the 2 chromosomes and swapping and interchanging them at that point forming 2 new chromosomes. Refer to table 1 for how this project implemented crossover for GA.

Mutation Operator – Using the children chromosomes generated in the previous step, we applied mutation to maintain the genetic diversity in each generation the generations. In order to increase the diversity in the next generations, we decided to have an adaptive mutation rate. This meant that the mutation probability increased after each iteration. This was achieved by obtaining the product of the mutation probability and a value calculated by . ~~A random value~~ between [0,1] was calculated, and if it was less than the mutation probability, each gene in the chromosomes were alternated. Refer to table 2 for how this project implemented mutation for GA.

Termination – The termination criteria was a max number of terminations. The algorithm stops after 20 iterations.

The pseudocode for how we implemented GA is as follows:

Initialize Parameters: PopulationSize =20, W=0.9, CR=0.6

Loop 1: Repeat until Count = 10

1) Randomly initialize population p

Loop 2: Repeat until Count = 20

2) Determine fitness of current population using the fitness function and rank chromosomes

3) Return best solution and update GBest(global best) if required

4) Select 2 parents from previous population using the roulette wheel selection

5) Apply the Crossover operator if random < CR and generate the next generation

6) Calculate the Mutation Probability

7) Perform the Mutation operator if required forming the next generation

8) Store Gbest value and the solution associated with it End Loop 1

9) Return the best solution from the 10 stored and the get selected features for training and testing in the DNN

4.2. PSO

The following is how we implemented PSO for feature selection.

Particle representation - Each particle is represented by a binary list consisting of 8 elements. Each element represents the selection of a specific diagnostic measurement. If the element is 1, the diagnostic measurement is chosen to train the neural network, if the latter binary digit 0, then it is not chosen to train the neural network with.

Fitness Function – To measure the optimality of a solution using a loss function which returned a value between [0,1].This accuracy is calculated using KNN with neighbour size 8.We then calculate FR which is the number of features chosen for a particle divided by the total number of features

which is 8.We then have w which is an arbitrary number between 0 and 1 which we set to 0.9. Due to the fitness function being a loss function, the lower the fitness, the better the solution.

The equation is as follows:

$$=(*)+(1-)* \quad (3)$$

Where:

- 01
- =(1− h h 5)
- =

The population consists of particles. We want each particle to have at most 6 features selected, so to not defeat the purpose of choosing less features. We don't want all 8 features selected. To do this we generate a random number between 1 and 6. This number will be the number of features that “could” be selected. Not necessarily selected. This number will represent the size of a list. Each element in the list will represent an index of a feature in the particle we want to set 1, indicating that feature has been selected. So we randomly generate indexes that are in the same range as the particle. I.e. between 0-7 as there are 8 features. The index within the particle which is not in the list gets a 0 value.

We iterate 30 times to move the particles within our swarm to a new position based on their personal and global best.

If the current generation is not the first generation, then we repeat steps 2, 3, 4 in the pseudocode below, as steps 5, 6, 7.

We update as:

$$=- (/)*(-) \quad (4)$$

Refer to equation (1) and (2) for how we update velocity. Essentially we subtract the particle's personal best solution pbest (VECTOR) from the particle and we subtract the global best particle gbest(VECTOR) from the particle.

After the update to the particle's position which was based on velocity, we create two new particles. The one with the better fitness will replace the current particle. To do this we use a squashing function:

$$1 - \frac{1}{1+} < 0$$

$$=(\quad \frac{1}{1+^{-}} \quad \geq 0$$

We iterate through the particle and get the value of each feature and use the squashing function to return the value between 0 and 1. If >0.5, you set that feature to 1 otherwise set it to 0. We then have two new particles with features either selected or not selected. We then calculate the fitness of these particles. And the particle with the better fitness will replace the current particle.

This ends the 30 iteration and is repeated 10 times. After which we obviously have our global best particle and we output that particle. And select only columns from the PIMA dataset with those features and feed it to the neural network.

The pseudocode for PSO that we used is:

Initialize Parameters: = ., PopulationSize=20, MaxIteration=30, C1=2, C2=22, max =0.9, Min=0.4

Loop 1: Repeat until Count = 10

1) Initialize each particle with at most 6 features with random velocity

2) Calculate the fitness of each particle

3) Updateif currentFitness > pbestFitness

4) Choose the particle with the highest accuracy as gbest

5) Calculate the fitness of each particle

6) Updateif currentFitness > pbestFitness

7) Update gbest if currentFitness> gbestFitness

8) Update

9) Update velocity

10) Update position based on velocity

11) Store and output gbest value and the solution associated with it, along with its accuracy End Loop 1

12) Return the best solution from the 10 stored and the get selected features for training and testing in the DNN

4.3. PSO-GA

For PSO-GA, we are essentially keeping the PSO implementation as described above. We added 3 GA methods at the "end" of the PSO code to try and improve the PSO implementation.

Selection Method - We implemented elitism as our selection method, we select the top 30% of the population, and they will undergo crossover and mutation,

Crossover operator - The way we implemented crossover is the standard crossover method, where we choose a random crossover point, the first chromosome is created with the first gene/feature of the first particle up to the random crossover point and with the genes/features from the second particle from the random crossover point to the last gene/feature, the second particle is created with the first gene/feature of the second particle up to the random crossover point and with the genes/features from the first particle from the random crossover point to the last gene/feature. The best particle performs crossover with 2^{nd} best particle, and the 3^{rd} best particle performs crossover with the 4^{th} best particle and so on. Each crossover operation returns 2 chromosomes. Refer to table 2.

Mutation Operator - The mutation operator simply changes chooses a random gene, and it flips the value, if the gene is 1, then we change it to 0 and vice versa.

The pseudocode for PSO-GA that we used is:

Initialize Parameters: = ., PopulationSize=20, MaxIteration=30, C1=2, C2=22, max =0.9, Min=0.4

Loop 1: Repeat until Count = 10

1) Initialize each particle with at most 6 features with random velocity

2) Calculate the fitness of each particle

3) Updateif currentFitness > pbestFitness

4) Choose the particle with the highest accuracy as gbest

5) Calculate the fitness of each particle

6) Updateif currentFitness > pbestFitness

7) Update gbest if currentFitness> gbestFitness

8) Update

9) Update velocity

10) Update position based on velocity

11) Calculate the fitness of each particle

12) Updateif currentFitness > pbestFitness

13) Update gbest if currentFitness> gbestFitness

14) Sort the population based on their fitness function

15) Choose the best 30% of the population (Elitism)

16)Perform crossover with the elite particles with a random crossover point

17)Perform mutation on the elite particles

18) Calculate the fitness of each particle

19) Updateif currentFitness > pbestFitness

20) Update gbest if currentFitness> gbestFitness

End Loop 2

21) Store and output gbest value and the solution associated with it, along with its accuracy End Loop 1

22) Return the best solution from the 10 stored and the get selected features for training and testing in the DNN

---

5. Results and Discussion

For all our implementations we tested on two forms of data, processed data and unprocessed data. Processed data is where the 0 zero values are replaced with the mean value of that column and unprocessed data is the raw data. We ran each metaheuristic 10 times for feature selection. We saved the time of each iteration and the best features selected as well as the accuracy of that feature selection.

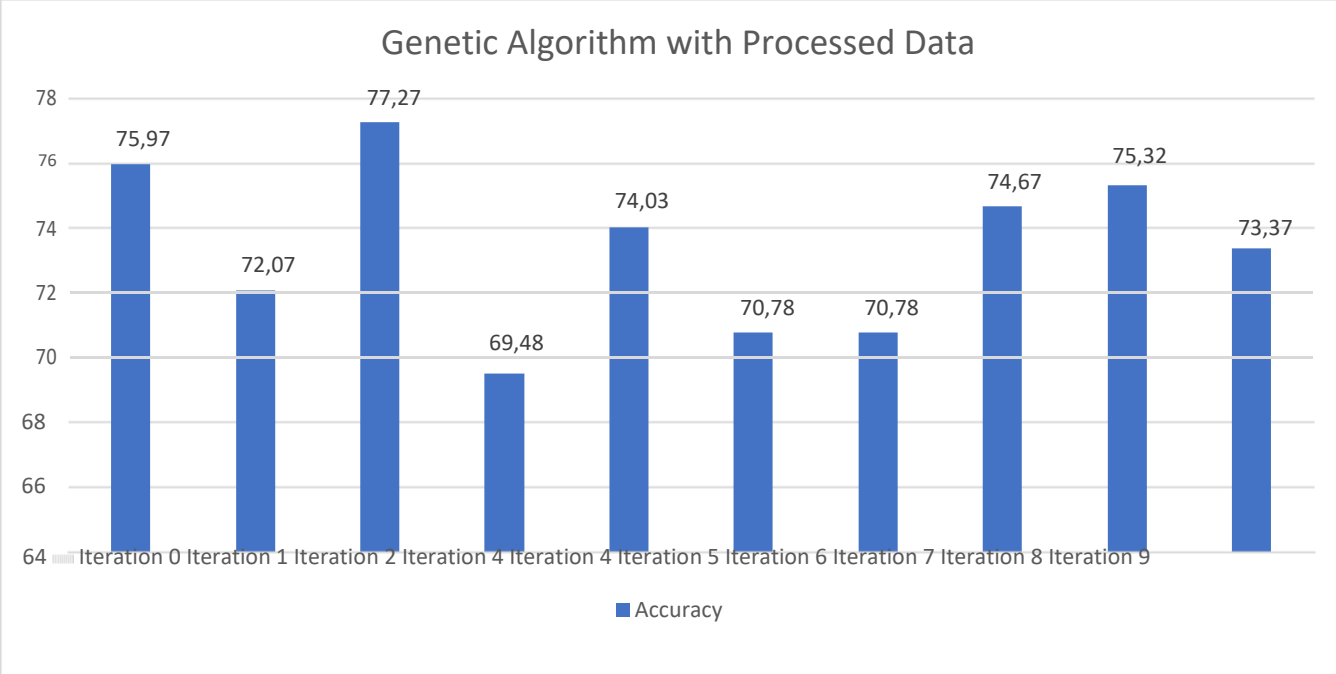
Below are the results for all our implementations.

Processed Data

GA

For the GA feature selection, these best results for the processed dataset came from the third iteration with accuracy 77%.

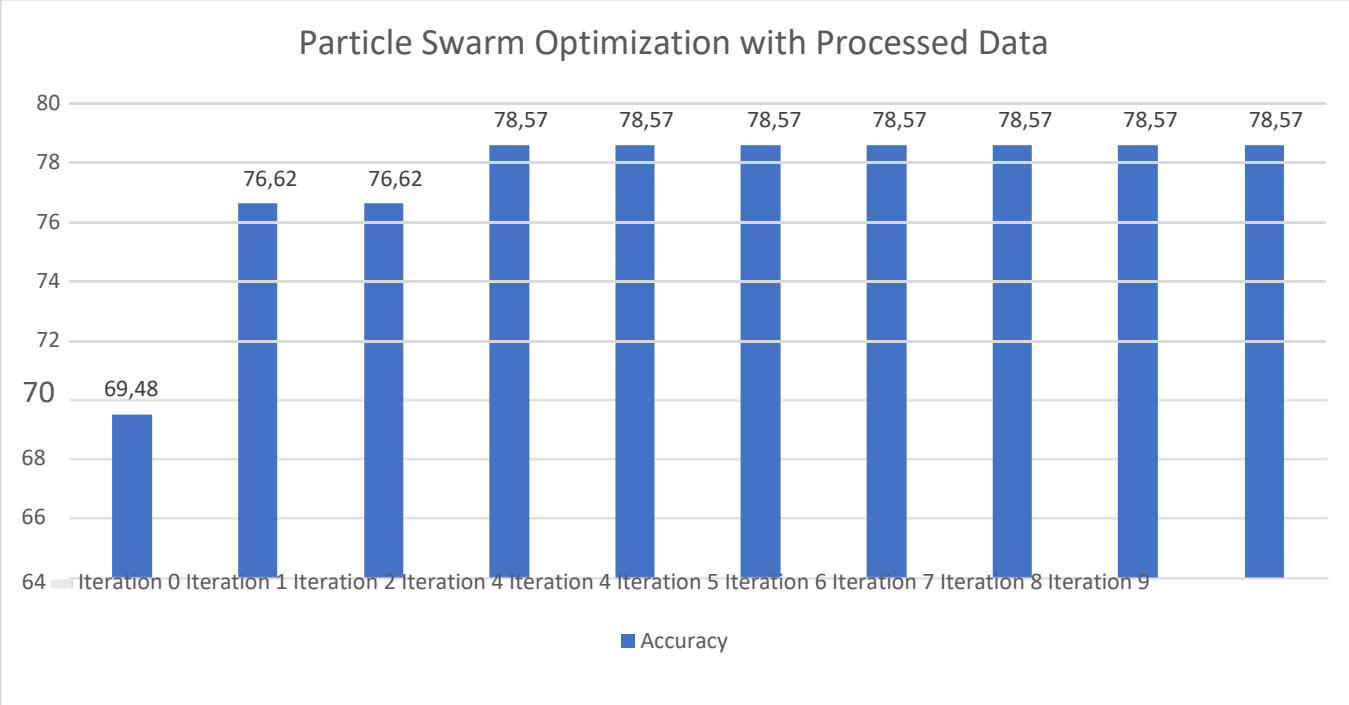
Best Accuracy	Best Vector Features	Best No Features	Time(seconds)
75,98%	[1. 1. 1. 1. 0. 0. 1. 1]	6	2,41
72,07%	[0. 1. 1. 1. 1. 0. 1. 1]	6	2.39
77,27%	[1. 0. 1. 0. 1. 1. 1. 0]	5	2.33
69,48%	[0. 1. 0. 0. 1. 0. 0. 1]	3	2..31
74,03%	[1. 1. 0. 1. 0. 0. 1. 0]	4	2.35
70,78%	[0. 1. 0. 1. 0. 0. 0. 0]	2	2.26
70,78%	[1. 1. 1. 0. 0. 1. 0. 0]	4	2.25
74,67%	[1. 1. 0. 1. 0. 1. 1. 1]	6	2.07
75,32%	[0. 0. 0. 1. 1. 1. 0. 0]	3	2.23
73,37%	[0. 1. 0. 1. 1. 0. 0. 0]	3	2.09



PSO

For the PSO feature selection, the best results were the same for iteration 4 to 10 with the same features selected and therefore same accuracy which was 78%. This is a clear example of how the PSO algorithm converges to a solution.

Best Accuracy	Best Vector Features	Best No Features	Time(seconds)
69,48%	[1, 0, 0, 0, 0, 0, 0, 0]	1	8,73
76,62%	[0, 1, 1, 1, 0, 0, 0, 0]	3	7,66
76,62%	[0, 1, 1, 1, 0, 0, 0, 0]	3	7,15
78,57%	[0, 1, 0, 0, 0, 1, 1, 0]	3	7,14
78,57%	[0, 1, 0, 0, 0, 1, 1, 0]	3	9,36
78,57%	[0, 1, 0, 0, 0, 1, 1, 0]	3	8,45
78,57%	[0, 1, 0, 0, 0, 1, 1, 0]	3	8,83
78,57%	[0, 1, 0, 0, 0, 1, 1, 0]	3	10,87
78,57%	[0, 1, 0, 0, 0, 1, 1, 0]	3	8,33
78,57%	[0, 1, 0, 0, 0, 1, 1, 0]	3	10,16

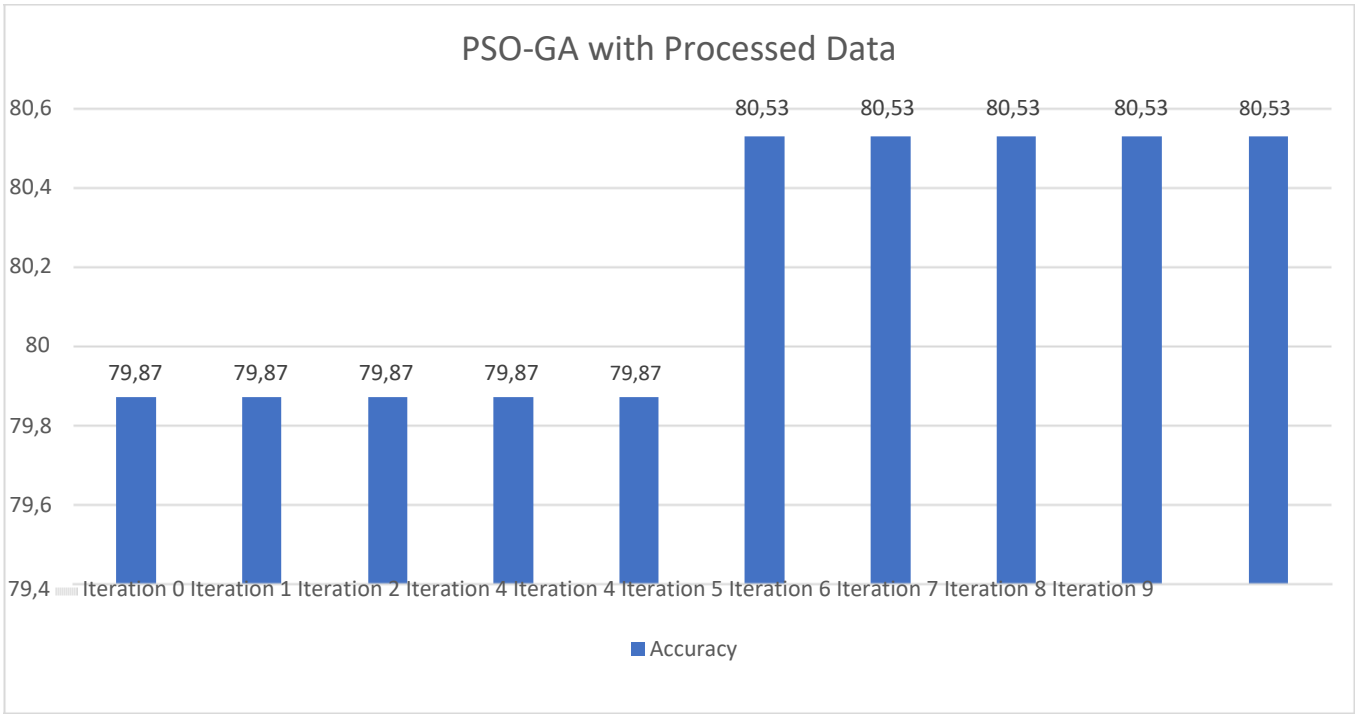


PSO-GA

For the PSO_GA feature selection, the best results were the same for iteration 6 to 10 with the same features selected and therefore same accuracy which was 80%.

Even though the PSO_GA algorithm converged, there is definite improvement in the optimal solution, and we attribute this as a success to the inclusion of GA.

Best Accuracy	Best Vector Features	Best No Features	Time(seconds)
79,87%	[1, 1, 0, 0, 0, 1, 1, 1]	5	16,39
79,87%	[1, 1, 0, 0, 0, 1, 1, 1]	5	14,94
79,87%	[1, 1, 0, 0, 0, 1, 1, 1]	5	15,73
79,87%	[1, 1, 0, 0, 0, 1, 1, 1]	5	15,43
79,87%	[1, 1, 0, 0, 0, 1, 1, 1]	5	15,01
80,53%	[1, 1, 0, 0, 0, 1, 1, 0]	4	15,54
80,53%	[1, 1, 0, 0, 0, 1, 1, 0]	4	14,48
80,53%	[1, 1, 0, 0, 0, 1, 1, 0]	4	14,74
80,53%	[1, 1, 0, 0, 0, 1, 1, 0]	4	15,21
80,53%	[1, 1, 0, 0, 0, 1, 1, 0]	4	16,34



Best Vectors and Selected Features

	GA	PSO	PSO_GA
Best feature vector:	[1. 0. 1. 0. 1. 1. 1. 0]	[0, 1, 0, 0, 0, 1, 1, 0]	[1, 1, 0, 0, 0, 1, 1, 0]
Selected Features:	Pregnancies, BloodPressure, Insulin, BMI, DiabetesPedigreeFunction	Glucose, BMI, DiabetesPedigreeFunction	Pregnancies, Glucose, BMI, DiabetesPedigreeFunction

Neural Network:

	GA	PSO	PSO_GA	Neural Network (no metaheuristic)
Total time(s):	129.53	126.65	119.23	146.14
Average Accuracy	72.40	76.69	77.02	75.40

From the results of the neural network where we tested all the features selected by each metaheuristic algorithm, we can see that the PSO_GA hybrid did select the best features based on the results with the accuracy of 77% which is 2% better than the neural network with all the features as well as just under 30 seconds faster.

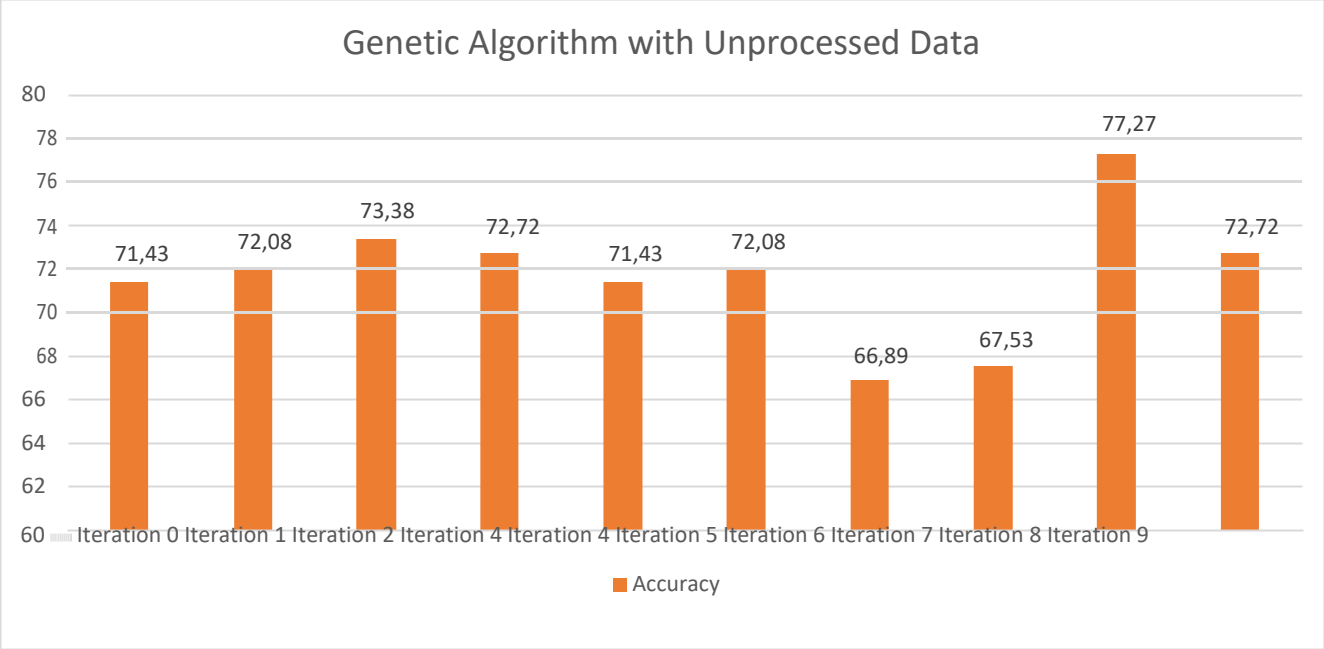
In this instance the features selected by GA did not perform better than the neural network with all the features used.

Unprocessed Data

GA Unprocessed Data

For the GA feature selection, these best results for the non-processed dataset came from the 9th iteration with accuracy 77.27%.

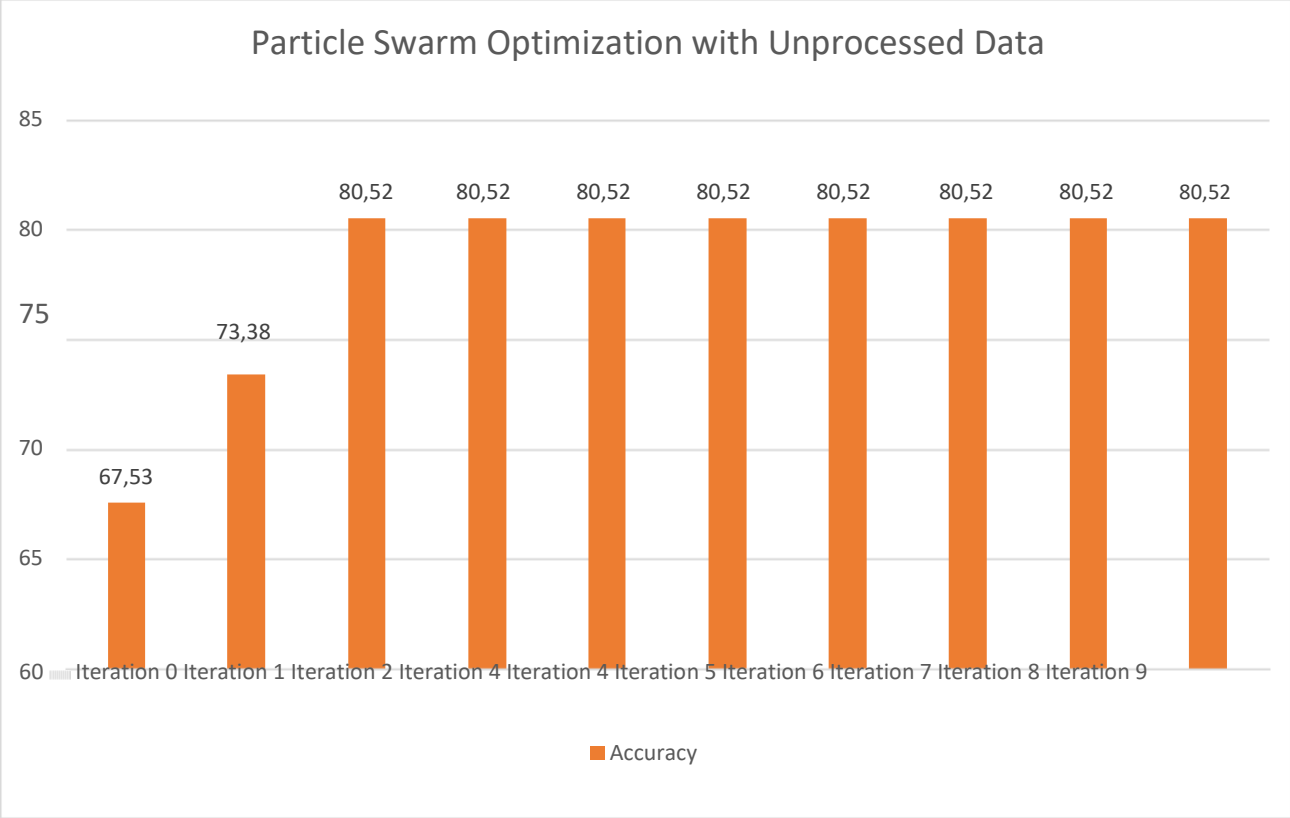
Best Accuracy	Best Vector Features	Best No Features	Time(seconds)
71,43%	[1. 1. 0. 0. 1. 1. 0. 1]	5	2.29
72,08%	[1. 1. 0. 0. 1. 0. 0. 0]	3	2.16
73,38%	[0. 1. 0. 1. 1. 0. 0. 0]	3	2.42
72,72%	[0. 1. 1. 0. 1. 0. 0. 1]	4	2.25
71,43%	[1. 0. 1. 0. 0. 1. 1. 0]	4	2.17
72,08%	[1. 1. 0. 0. 1. 0. 0. 0]	3	2.36
66,89%	[0. 0. 1. 0. 0. 0. 0. 1]	2	2.34
67,53%	[1. 1. 0. 0. 0. 0. 0. 0]	2	2.46
77,27%	[1. 0. 1. 0. 1. 1. 1. 0]	5	2.11
72,72%	[0. 1. 1. 0. 1. 0. 0. 1]	4	2.18



PSO

For the PSO feature selection, the best results were the same for iteration 3 to 10 with the same features selected and therefore same accuracy which was 80.52%. As shown in this in this table, this shows the shortcomings of PSO where it converges to a solution.

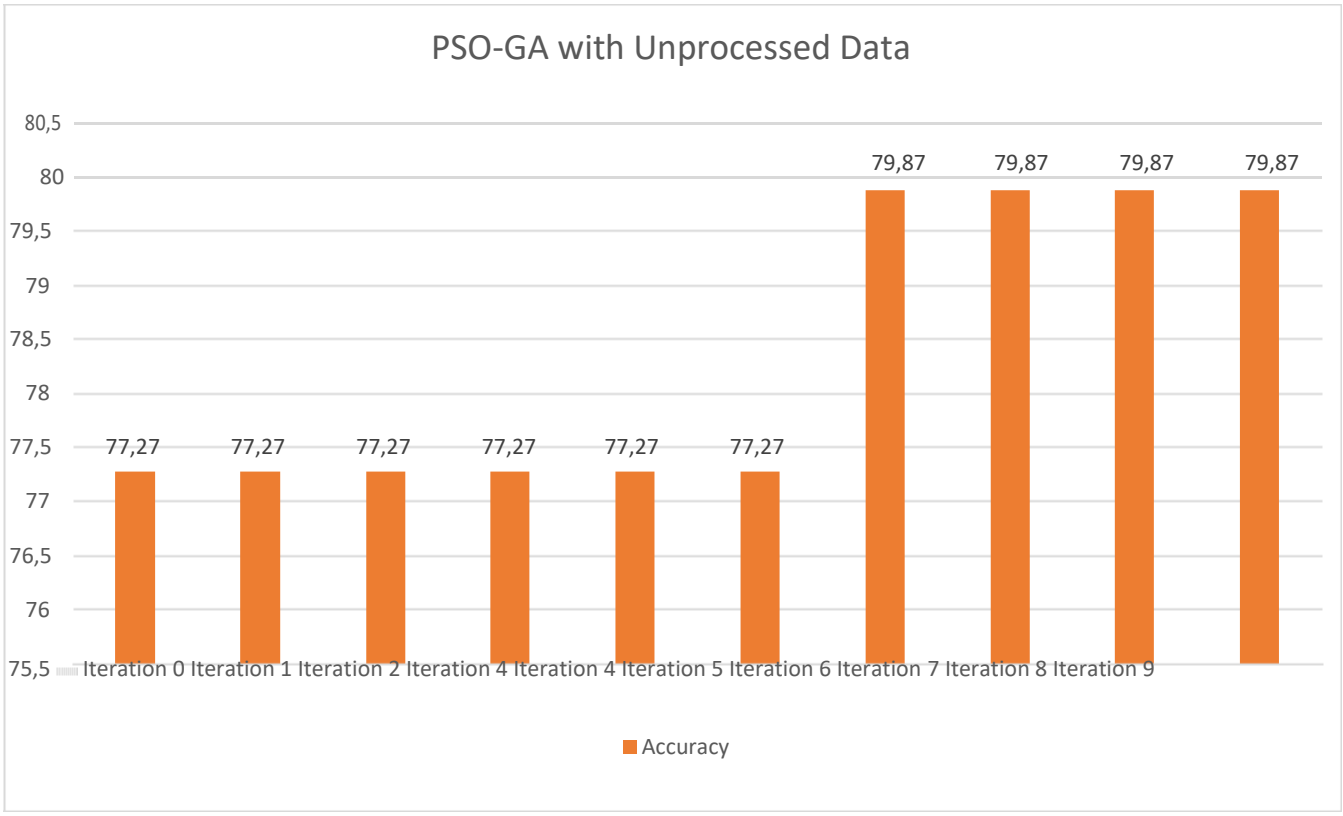
Best Accuracy	Best Vector Features	Best No Features	Time(seconds)
67,53%	[0, 0, 0, 0, 0, 1, 1, 0]	2	9,77
73,38%	[0, 1, 1, 0, 0, 0, 0, 0]	2	8,38
80,52%	[0, 1, 1, 0, 1, 1, 0, 0]	4	8,21
80,52%	[0, 1, 1, 0, 1, 1, 0, 0]	4	9,69
80,52%	[0, 1, 1, 0, 1, 1, 0, 0]	4	9,00
80,52%	[0, 1, 1, 0, 1, 1, 0, 0]	4	8,23
80,52%	[0, 1, 1, 0, 1, 1, 0, 0]	4	8,11
80,52%	[0, 1, 1, 0, 1, 1, 0, 0]	4	7,78
80,52%	[0, 1, 1, 0, 1, 1, 0, 0]	4	9,37
80,52%	[0, 1, 1, 0, 1, 1, 0, 0]	4	10,36



PSO-GA

For the PSO_GA feature selection, the best results were the same for iteration 7 to 10 with the same features selected and therefore same accuracy which was 79.87%. As before, the PSO_GA algorithm did converge but not as much as the PSO algorithm which was the main reason for combining GA with the PSO algorithm so that it would lower the chances of the algorithm converging.

Best Accuracy	Best Vector Features	Best No Features	Time(seconds)
77,27%	[0, 1, 0, 0, 0, 0, 1, 1]	3	15,22
77,27%	[0, 1, 0, 0, 0, 0, 1, 1]	3	14,81
77,27%	[0, 1, 0, 0, 0, 0, 1, 1]	3	14,92
77,27%	[0, 1, 0, 0, 0, 0, 1, 1]	3	15,19
77,27%	[0, 1, 0, 0, 0, 0, 1, 1]	3	15,64
77,27%	[0, 1, 0, 0, 0, 0, 1, 1]	3	15,43
79,87%	[0, 1, 1, 0, 0, 1, 0, 0]	3	17,32
79,87%	[0, 1, 1, 0, 0, 1, 0, 0]	3	17,66
79,87%	[0, 1, 1, 0, 0, 1, 0, 0]	3	15,90
79,87%	[0, 1, 1, 0, 0, 1, 0, 0]	3	15,80



Best Vectors and Selected Features

	GA	PSO	PSO_GA
Best feature vector:	[1. 0. 1. 0. 1. 1. 1. 0]	[0, 1, 1, 0, 1, 1, 0, 0]	[0, 1, 1, 0, 0, 1, 0, 0]
Selected Features:	Pregnancies, BloodPressure, Insulin, BMI, DiabetesPedigreeFunction	Glucose, BloodPressure, Insulin, BMI	Glucose, BloodPressure, BMI

Neural Network:

Just as we did for processed data, we ran each best selected features from the metaheuristic algorithms and the neural network with all features, and in this case the GA did the best with 74% which was 2% better than the neural network with all features and only 4 seconds faster. The neural network that ran tested the GA selected features were only 4 seconds faster.

	GA	PSO	PSO_GA	Neural Network
Total time(s):	123.25	133.07	119.18	127.06
Average Accuracy (%):	74.61	72.65	72.66	72.53

5.1. Computer Specifications

We trained and tested all our implementations on the same laptop. We used python 3.7, and we used Pycharm to run all the code.

- Installed RAM: 8GB
- Processor : Intel® Core™ i7-8550U CPU @ 1.80GHz
- Graphic Card: Radeon ™ 530

6. Conclusion

From the results, for the processed data, only the features selected by the GA did not perform better than the neural network with all the features selected. We show a 2% increase in accuracy and 30 seconds faster neural network with features selected with our PSO-GA hybrid. We can see a definite importance in whether the data has been processed or not, for the un-processed data we can see all the neural networks tested decreased in value except for GA.

We can conclude that the process of feature selection is important in any machine learning process, and it evidently has applications in the medical industry. We can see a clear increase in accuracy for the neural network and less run time when using PSO and PSO-GA hybrid for feature selection with processed data. The results for the unprocessed data show the best GA features did not contain glucose as a feature, and it increased in accuracy for the neural network. Glucose is essential in classifying diabetes, either the GA detected the features required to classify diabetes without glucose or the unprocessed data had such a major impact that it undermined the importance of glucose in diabetes.

For future work, we suggest using different metaheuristic algorithms for feature selection. We also suggest using metaheuristic algorithms for the training of a deep neural network, this would essentially create and train the weights of a deep neural network. The overall aim would be to increase accuracy as well as decreasing run time.

7. References

[1] J. Sun, "The Study of Pima Indian Diabetes", 2020. Available: 10.13140/RG.2.2.14751.76960 [Accessed 12 July 2020].

[2] Choubey, D.K., Paul, S., Kumar, S. and Kumar, S., 2017, February. Classification of Pima indian diabetes dataset using naive bayes with genetic algorithm as an attribute selection. In Communication and Computing Systems: Proceedings of the International Conference on Communication and Computing System (ICCCS 2016) (pp. 451-455).

[3] A. Pietrangelo and K. Cherney, "The Effects of Diabetes on Your Body", *Healthlne*, 2020. [Online]. Available: https://www.healthline.com/health/diabetes/effects-on-body#2. [Accessed: 12- Jul- 2020].

[4] "International Diabetes Federation - Facts & figures", *Idf.org*, 2020. [Online]. Available: https://www.idf.org/aboutdiabetes/what-is-diabetes/facts-figures.html. [Accessed: 12- Jul- 2020].

[5] B. Lee and J. Kim, "Identification of Type 2 Diabetes Risk Factors Using Phenotypes Consisting of Anthropometry and Triglycerides based on Machine Learning", *IEEE Journal of Biomedical and Health Informatics*, vol. 20, no. 1, pp. 39-46, 2016. Available: 10.1109/jbhi.2015.2396520 [Accessed 13 July 2020].

[6] I. Kavakiotis, O. Tsave, A. Salifoglou, N. Maglaveras, I. Vlahavas and I. Chouvarda, "Machine Learning and Data Mining Methods in Diabetes Research", Computational and Structural

Biotechnology Journal, vol. 15, pp. 104-116, 2017. Available: 10.1016/j.csbj.2016.12.005 [Accessed 13 July 2020].

[7] M. Alghamdi, M. Al-Mallah, S. Keteyian, C. Brawner, J. Ehrman and S. Sakr, "Predicting diabetes mellitus using SMOTE and ensemble machine learning approach: The Henry Ford ExercIse Testing (FIT) project", PLOS ONE, vol. 12, no. 7, p. e0179805, 2017. Available: 10.1371/journal.pone.0179805 [Accessed 13 July 2020].

[8] S. I. Ayon, and M. M. Islam, "Diabetes prediction: A deep learning approach," Int. J. Inf. Eng. Electron. Bus., Vol. 11, no. 2, pp. 21–7, Mar. 2019.

[9] F. Beloufa and M. Chikh, "Design of fuzzy classifier for diabetes disease using Modified Artificial Bee Colony algorithm", *Computer Methods and Programs in Biomedicine*, vol. 112, no. 1, pp. 92-103, 2013. Available: 10.1016/j.cmpb.2013.07.009 [Accessed 13 July 2020].

[10] S. AlMuhaideb and M. Menai, "HColonies: a new hybrid metaheuristic for medical data classification", *Applied Intelligence*, vol. 41, no. 1, pp. 282-298, 2014. Available: 10.1007/s10489-014-0519-z [Accessed 13 July 2020].

[11] M. Daliri, "Feature selection using binary particle swarm optimization and support vector machines for medical diagnosis", *Biomedizinische Technik/Biomedical Engineering*, vol. 57, no. 5, 2012. Available: 10.1515/bmt-2012-0009 [Accessed 13 July 2020].

[12] K. Ateeq and D. Ganapathy, "The novel hybrid Modified Particle Swarm Optimization – Neural Network (MPSO-NN) Algorithm for classifying the Diabetes", *International Journal of Computational Intelligence Research*, vol. 13, no. 4, pp. 595-614, 2017. [Accessed 13 July 2020].

[13] H. Sahebi and S. Ebrahimi, "A Fuzzy Classifier Based on Modified Particle Swarm Optimization for Diabetes Disease Diagnosis", *Advances in Computer Science : an International Journal*, vol. 4, no. 3, pp. 11-17, 2015. Available: http://www.acsij.org/acsij/article/view/90. [Accessed 13 July 2020].

[14] E. Sreedevi and M. Padmavathamma, "A Threshold Genetic Algorithm for Diagnosis of Diabetes using Minkowski Distance Method", 2015. [Accessed 15 July 2020].

[15] J. Holland, "Genetic Algorithms and Adaptation", *Adaptive Control of Ill-Defined Systems*, pp. 317-333, 1984. Available: 10.1007/978-1-4684-8941-5_21 [Accessed 14 July 2020].

[16] J. Holland, *Adaptation in natural and artificial systems*. Cambridge, Mass.: MIT Press, 1992.

[17] S. Chatterjee and A. Bhattacherjee, "Genetic algorithms for feature selection of image analysis-based quality monitoring model: An application to an iron mine", *Engineering Applications of Artificial Intelligence*, vol. 24, no. 5, pp. 786-795, 2011. Available: 10.1016/j.engappai.2010.11.009 [Accessed 14 July 2020].

[18] W. Siedlecki and J. Sklansky, "A note on genetic algorithms for large-scale feature selection", *Pattern Recognition Letters*, vol. 10, no. 5, pp. 335-347, 1989. Available: 10.1016/0167-8655(89)90037-8 [Accessed 14 July 2020].

[19] B. Wutzl, K. Leibnitz, F. Rattay, M. Kronbichler, M. Murata and S. Golaszewski, "Genetic algorithms for feature selection when classifying severe chronic disorders of consciousness", *PLOS ONE*, vol. 14, no. 7, p. e0219683, 2019. Available: 10.1371/journal.pone.0219683 [Accessed 15 July 2020].

[20] S. Li, H. Wu, D. Wan and J. Zhu, "An effective feature selection method for hyperspectral image classification based on genetic algorithm and support vector machine", *Knowledge-Based Systems*, vol. 24, no. 1, pp. 40-48, 2011. Available: 10.1016/j.knosys.2010.07.003 [Accessed 15 July 2020].

[21] S. Sayed, M. Nassef, A. Badr and I. Farag, "A Nested Genetic Algorithm for feature selection in high-dimensional cancer Microarray datasets", *Expert Systems with Applications*, vol. 121, pp. 233-243, 2019. Available: 10.1016/j.eswa.2018.12.022 [Accessed 15 July 2020].

[22] X. Yang, Nature-Inspired Optimization Algorithms. Elsevier, 2014.

[23] S. Shah, "Genetic Algorithm for a class of Knapsack Problems", 2019. [Accessed 4 June 2020].

[24] R. Singh, "Solving 0–1 Knapsack problem using Genetic Algorithms", 2011. Available: doi: 10.1109/ICCSN.2011.6013975 [Accessed 5 June 2020].

[25] J. Kennedy and R. Eberhart, "Particle swarm optimization", *Proceedings of ICNN'95 - International Conference on Neural Networks*. Available: 10.1109/icnn.1995.488968 [Accessed 16 July 2020].

[26] V. Kachitvichyanukul, "Comparison of Three Evolutionary Algorithms: GA, PSO, and DE", *Industrial Engineering and Management Systems*, vol. 11, no. 3, pp. 215-223, 2012. Available: 10.7232/iems.2012.11.3.215 [Accessed 17 July 2020].

[27] B. Xue, M. Zhang and W. Browne, "Particle Swarm Optimization for Feature Selection in Classification: A Multi-Objective Approach", *IEEE Transactions on Cybernetics*, vol. 43, no. 6, pp. 1656-1671, 2013. Available: 10.1109/tsmcb.2012.2227469 [Accessed 16 July 2020].

[28] Y. Zhang, D. Gong, X. Sun and Y. Guo, "A PSO-based multi-objective multi-label feature selection method in classification", *Scientific Reports*, vol. 7, no. 1, 2017. Available: 10.1038/s41598-017-00416-0 [Accessed 17 July 2020].

[29] H. Inbarani, A. Azar and G. Jothi, "Supervised hybrid feature selection based on PSO and rough sets for medical diagnosis", *Computer Methods and Programs in Biomedicine*, vol. 113, no. 1, pp. 175-185, 2014. Available: 10.1016/j.cmpb.2013.10.007 [Accessed 17 July 2020].

[30] S. Vieira, L. Mendonça, G. Farinha and J. Sousa, "Modified binary PSO for feature selection using SVM applied to mortality prediction of septic patients", *Applied Soft Computing*, vol. 13, no. 8, pp. 3494-3504, 2013. Available: 10.1016/j.asoc.2013.03.021 [Accessed 17 July 2020].

[31] Z. Abdmouleh, A. Gastli, L. Ben-Brahim, M. Haouari and N. Al-Emadi, "Review of optimization techniques applied for the integration of distributed generation from renewable energy sources", *Renewable Energy*, vol. 113, pp. 266-280, 2017. Available: 10.1016/j.renene.2017.05.087 [Accessed 17 July 2020].

[32] Y. Zhang, S. Balochian, P. Agarwal, V. Bhatnagar and O. Housheya, "Artificial Intelligence and Its Applications", *Mathematical Problems in Engineering*, vol. 2014, pp. 1-10, 2014. Available: 10.1155/2014/840491 [Accessed 17 July 2020].

[33] Y. Zhang, S. Wang and G. Ji, "A Comprehensive Survey on Particle Swarm Optimization Algorithm and Its Applications", *Mathematical Problems in Engineering*, vol. 2015, pp. 1-38, 2015. Available: 10.1155/2015/931256 [Accessed 17 July 2020].

[34] H. Garg, "A hybrid PSO-GA algorithm for constrained optimization problems", *Applied Mathematics and Computation*, vol. 274, pp. 292-305, 2016. Available: 10.1016/j.amc.2015.11.001 [Accessed 17 July 2020].

[35] X. Shi, L. Wan, H. Lee, X. Yang, L. Wang and Y. Liang, "An improved genetic algorithm with variable population-size and a PSO-GA based hybrid evolutionary algorithm", *Proceedings of the 2003 International Conference on Machine Learning and Cybernetics (IEEE Cat. No.03EX693)*, 2003. Available: 10.1109/icmlc.2003.1259777 [Accessed 18 July 2020].

[36] M. Nazir, A. Majid-Mirza and S. Ali-Khan, "PSO-GA Based Optimized Feature Selection Using Facial and Clothing Information for Gender Classification", *Journal of Applied Research and Technology*, vol. 12, no. 1, pp. 145-152, 2014. Available: 10.1016/s1665-6423(14)71614-1 [Accessed 18 July 2020].

[37] S. Khan, M. Nazir, N. Riaz and M. Khan, "Optimized Features Selection using Hybrid PSO-GA for Multi-View Gender Classification", *International Arab Journal of Information Technology*, vol. 12, no. 2, pp. 183-189, 2013. [Accessed 18 July 2020].