CAPSTONE PROJECT 2

Final Year Project Report

An Artificial Intelligence Model for Flood Prediction in Urban Areas Based on Swarm Intelligence Algorithms and Artificial Neural Networks

by

Chew Chien Zhen

19037746

Bachelor of Software Engineering (HONS)

Supervisor: Dr. Muhammed Basheer Jasser

Semester: April 2023

Date: 18 July, 2023

Department of Computing and Information Systems

School of Engineering and Technology

Sunway University

Table of content

Abstract

In this phase second phase of my capstone project, the main objective would be focusing on the implementation of multiple algorithms into my flood prediction models with a goal of reaching a more optimized output.

An example of an optimized output would be having a higher accuracy since my capstone topic is related to flood predictions therefore the output would be the accuracy of the flood predictions. Therefore, the multiple attempts that will take place in this capstone project will also be recorded and documented for documentation purposes and materials for analysis. The programming language of the existing work which will be used in this phase of my capstone project is done in the Python language therefore my attempts of optimization will also be in the Python language.

Floods are the most frequently occurrence that has been ongoing as long as history could tell and still is prevailing in bringing drastic damage to the environment, economic and social aspect. According to the World Health Organization (WHO), natural disasters have affected as many as 150 million people annually. By the year 2020, it is estimated that a total of 268$ billion was calculated from the damage of natural disasters and floods was a huge contributor to the damage cost. WHO also mentioned that between the year 1998 to 2017, more than 2 billion in the world population was affected by flood.

1.1 Problem statement

Among all natural disasters recorded in history, flood is known to be one of the most destructive, causing severe damage at a large scale to mankind be it in the socioeconomic systems, agriculture, and infrastructure. To top off the severity of the destructive capabilities, the damage can also be done in a very short amount of time which is also the main culprit of the rate of death during the natural disaster taking place. Governments in different countries are under the pressure of developing maps of flood risks urban areas with high accuracy for reliable planning and sustainability management in preparation for protecting, preventing, and preparation for any floods.

Flood prediction models are essentials when it comes to the assessment of hazards and the extreme event of causes management. Thorough and accuracy in the prediction processes are game changing to water recourse management strategies, emergency evacuation modelling, and policy analysis and suggestions. Therefore, the urgency of having advanced systems for long-term and short-term flood predictions and any other natural events that involves water is very much considered to minimize damage as much as possible. (Mosavi & Ozturk & Chou, 2018). Besides, climatic changes have its own affects as the frequency of rainfall occurrences potentially enhances the rate of flooding. Many resides in proximity with flood potential areas such rivers therefore precautions and safety measures are a requirement to reduce casualties and other risks associated alongside. Flood predictions are a very complicated task even for experts in this field of studies due to the uncertainties. Computational algorithms such as the neural networks has been used frequently to estimate flood in potential areas like a river and its effects on its outer surroundings, for instants, the flow that is upstream of the river is very useful when looking for the flow of downstream since downstream flow usually lacks the measurements (Zehra, n.d).

Modern times major flood predictions are modelled mainly from data specifications, and it involves many simplified assumptions. Therefore, to simulate similar complex mathematical expressions which signifies the processes that are physical as well as basin behaviours. Models like these benefits from techniques that are specific to the process models. Examples are stochastic, deterministic, continuous, hybrid and many more.

1.2 Objectives

The objective of this project is to better understand the significance of flood prediction using machine learning and how to optimize it for better results. The main purpose of this is to minimize the damage dealt by flood disasters to the environment, communities, and economy. The objectives are as follow:

- Study existing works that are related to flood prediction using Swarm Intelligence and Artificial Neural Network.
- Compare the related works and its results of the predictions thoroughly.
- Implement an algorithm(s) into the methods as an optimization procedure.
- Evaluate and test the output of the optimized work.

1.3 Scope of Work

To study how flood predictions can be predicted and recorded using machine learning algorithms. The studies done will be evaluate as the following steps.

During the start of the work stage, a document gathering all the related works regarding flood prediction using machine learning. The works will then be thoroughly investigated and studied to better know the methods used in the work done on flood predictions. The follow up would be thoroughly analysing the dataset gathered on the studied of urban areas.

The data gather can be used to evaluate flood risks predictions which needs accurate spatial, some temporal information to minimize damage loss. (Bui et al, 2020). As to how floods are predicted, it is through the usage of machine learning algorithms that are implemented with the inputs of the dataset and the output would be an accuracy evaluation alongside the Root Mean Square Error (RMSE) and the Mean Absolute Error (MAE) as the results of this work. These outputs were used to predict the accuracy of the predictions of floods. Moving back to dataset, the areas of studies in obtaining the data for the inputs are focused around areas that are prone to flash floods like tropical countries which are one of the most vulnerable to disasters due to climate changes that happens in a short period of time alongside a high frequency. (Bui et al, 2020).

The work that will be conducted throughout the phases of this project would be to obtain an already existing dataset and code of the flood predictions and optimize the work for a better output of readings. The optimization must use swarm intelligence or artificial neural network algorithms as an acceptance criterion in this project scope. The language that will be used in this optimization experiments will the Python language since most optimization in swarm intelligence that was gathered throughout the studies are python approached optimizations therefore python libraries such as Pandas and NumPy will be used in the optimization process of existing works. Other tools like Visual Studio Code and PyCharm will be used in the experimentation optimization process as these tools have built in extensions that may be useful for the experimental phase in capstone 2 down the line.

Finally, the outputs will be determined and evaluated based off its accuracy and the results will be documented to be analysed alongside existing results of other works gathered in thus project. Along the process of experiments, failed attempts and results will also be recorded altogether for evaluations in errors from the steps taken which produced such failures in results which will be done in capstone 2 of this project.

2.0 Literature review

The purpose of this literature review section is to study and analyse the related work to flood predictions using machine learning and how modern and advanced techniques are used as well as methods in forecasting flood dangers which allows countermeasures. This literature review will review some the works that are done and proven effective in forecasting floods as well as highlighting the few algorithms used in the work process. After that, a comparison will be done based off the related works gathered and will be compiled together into a form of discussion of the related works.

2.1 Introduction

This section will discuss mainly on some of the known factors and background of Artificial Intelligence, Swarm Intelligence Algorithms, Artificial Neural Networks and Flood prediction. The follow up after would be the problem statement of this project which is then followed up by literature reviews and then a section for related works and discussions. Then, methodologies of optimization algorithms will be presented in this report alongside datasets and an experimental history and failed results will be tabulated in this report. After that, a comparison of the related works is done, and a self-discussion will be made to determine the algorithm that will be used in the optimization process in the next phase of this project which is capstone 2

2.2 Backgrounds

Artificial Intelligence (AI)

Just less than a decade after helping the Allied Forces win World War II by breaking the Nazi encryption machine Enigma, mathematician Alan Turing revolutionized the world of technology with a simple question: “Can machines think?” To follow up, Turing’s paper “Computing Machinery and Intelligence” in 1950 with its subsequent Turing Test gave shape to the fundamental goal and vision of AI. At the very centre, AI serves as the branch in the world of computer science whose purpose is to find answers for Turing’s revolutionary question. It is the very process journey of replicating and simulating the human mind and its intelligence, all inside a machine. Such a goal which in expands in continuation, has left many questions and debates so much that not a single definition in this field will ever be universally accepted in present era of knowledge (Schroer, 2022).

Artificial intelligence, commonly known for short as A.I, is often depicted as a machine that can replicate human’s learning, reasoning, perception, problem-solving and language (Bradford, 2022). The AI computer is programmed to have the capability to “think” like a human being and use what it has learned and start construction on the information they have learned without any human interventions. Therefore, such capabilities allow the computer to adapt with foreign and new situations, like how Siri remembers which music you like and use that knowledge to suggest other new music based on what you like (Bradford, 2022).

As it relates to artificial intelligence, there are several types of learning. The most basic method is trial and error. A simple computer programme for solving mate-in-one chess problems, for example, might try random moves until mate is found. The programme may then save the solution along with the position, so that the next time the computer encounters the same position, it will remember the solution. This rote learning of individual items and procedures is relatively simple to implement on a computer. The problem of implementing what is known as generalisation is more difficult. Generalization is the process of applying previous experience to similar new situations. (Copeland, 2022)

<u>**Swarm Intelligence (SI)**</u>

In the year 1989, Swarm Intelligence, also commonly known as SI, was introduced by Gerardo Beni and Jing Wang. S.I has a simple meaning: by using knowledge of a group of organisms commonly known as collective objects. Together, the collective objects data is used to reach the most optimized solution when solving a problem. The term “Swarm” refers to a group of objects be it humans, animals, etc. An example would be, giving a problem to a group of students (the swarm) and ask the group to reach the best possible solution among the members of the group. By using the knowledge of the group, an optimized solution was procured where the solution is something that every member agrees in the group. (geeksforgeeks.org, 2021)

In the world of robotics, swarm intelligence is used by scientists and applying what the scientists have observed from nature to the machines. For instants, a swarm consisting of drones equipped with sensors on each of them. When information is collected by the drones through the sensor, one drone will share the information they found to the other drones in the group(swarm) which allows the group of drones to function as one.

Swarm intelligence (SI) is a technique of computational intelligence used to solve complex problems. SI entails a group study of how individuals in a population interact with one another on a local level. Nature is a great source of inspiration, especially for biological systems. Agents follow simple rules and there is no centralised control structure in place to predict individual agent behaviour. The random iteration of a certain degree between the agents results in "intelligent" behaviour that is unknown to individual agents. It is also a branch of Computer Intelligence that was developed by simulating the collective intelligent behaviour of insect or animal groups such as flocks of birds, ant colonies, schools of fish, and bee swarms. Hence the algorithm names like Artificial Bee Colony (ABC), Ant Colony Optimization (ACO), Grey-Wolf Optimization (GWO) and many more. (ScienceDirect, 2020)

Examples of work done with Swarm Intelligence

The pattern of x of N input with the variables $\boldsymbol{x} \in R^N$, **Extreme Learning Machine (ELM)** output with L hidden nodes and q out nodes.

$$f_L(\boldsymbol{x}) = \sum_{i=1}^{L} \beta_i G(IW_{\boldsymbol{i}}, BA_{\boldsymbol{i}}, \boldsymbol{x})$$

Figure 1.1, Equation of **Extreme Learning Machine (ELM)** extracted from *A novel hybrid approach based on a swarm intelligence optimized extreme learning machine for flash flood susceptibility mapping*

The output matrix is represented by H where EL; $\boldsymbol{x}_1$, $\boldsymbol{x}_2$, …, $\boldsymbol{x}_N$ which are the flood conditions factors; and T represents the training target values of the dataset.

$$OW = \boldsymbol{H} \dagger \boldsymbol{T}$$

where **H†** is the Moore–Penrose generalized inverse of **H**.

$$\boldsymbol{H} = \begin{bmatrix} \boldsymbol{h}(\boldsymbol{x}_1) \\ \vdots \\ \boldsymbol{h}(\boldsymbol{x}_N) \end{bmatrix} = \begin{bmatrix} G(\boldsymbol{a}_1, b_1, \boldsymbol{x}_1) & \dots & G(\boldsymbol{a}_L, b_L, \boldsymbol{x}_1) \\ \vdots & \dots & \vdots \\ G(\boldsymbol{a}_1, b_1, \boldsymbol{x}_N) & \dots & G(\boldsymbol{a}_L, b_L, \boldsymbol{x}_L) \end{bmatrix} N \times L$$

$$\beta = \begin{bmatrix} \beta_1^T \\ \vdots \\ \beta_L^T \end{bmatrix} N \text{ x } q \quad \text{and } \boldsymbol{T} = \begin{bmatrix} \boldsymbol{t}_1^T \\ \vdots \\ \boldsymbol{t}_L^T \end{bmatrix} N \times q$$

Figure 1.2, Equation continuation of **Extreme Learning Machine (ELM)** extracted from *A novel hybrid approach based on a swarm intelligence optimized extreme learning machine for flash flood susceptibility mapping*

The *i*th particle expressed as the vectors of representing the velocity, p identifies the best locations of its neighbours.

$$\boldsymbol{V_i} \leftarrow \chi \left(\boldsymbol{V_i} + \varphi \left(\left(\sum_{j=1}^{K_i} a_j \boldsymbol{P}_{nbrj} \right) - X_i \right) \right)$$

Figure 1.3, Equation of **Particle Swarm Optimization (PSO)** extracted from *A novel hybrid approach based on a swarm intelligence optimized extreme learning machine for flash flood susceptibility mapping*

Neighbour is measured with generalized random coefficient, a_j and U ranges from 0 to 1 which signifies a uniform random number

$$a_j = \frac{b_j}{\sum b_j}, bj = U\,(0,1), \mathrm{j} = 1, \ldots, Ki$$

Figure 1.4, Equation continuation of **Particle Swarm Optimization (PSO)** extracted from *A novel hybrid approach based on a swarm intelligence optimized extreme learning machine for flash flood susceptibility mapping*

Following is a formulation of how grasshopper swarms in nature behave when looking for food sources:

$$X_i\,(t+1) = c\left(\sum_{\substack{j=1\\ j\neq i}}^{N} c\frac{ub_d - lb_d}{2} s\left(\left|X_j\,(t) - X_{i(t)}\right|\right)\frac{x_j(t)-x_i(t)}{d_{ij}}\right) + \hat{T}_d$$

Figure 1.5, Equation of **Grasshopper Optimization Algorithm (GOA)** extracted from *Flood spatial prediction modelling using a hybrid of meta-optimization and support vector regression modelling*

Xi(t + 1) signifies the position of the i^{th} grasshopper at t + 1 and c is the reduction coefficient to balance out the timing of the exploration and exploitation stages. c is expressed as:

$$c\,(t) = c_{\max} - t\frac{c_{\max} - c_{\min}}{t_{\max}}$$

Figure 1.6, Continuation equation of **Grasshopper Optimization Algorithm (GOA)** extracted from *Flood spatial prediction modelling using a hybrid of meta-optimization and support vector regression modelling*

C.max and *C*.min respectively signifies maximum and minimum values in the parameter of c(t) while t and t.max both represents the maximum number of iterations.

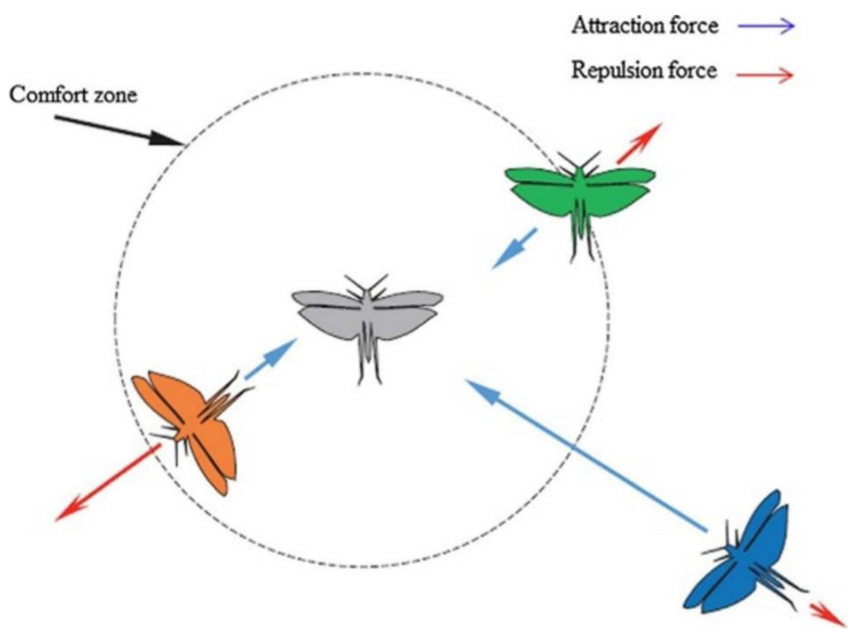


Figure 1.7, **Grasshopper Optimization Algorithm (GOA)** extracted from *Flood spatial prediction modelling using a hybrid of meta-optimization and support vector regression modelling*

The probability option which is $L_{i(j)}$ is a chosen path at the cycle *k* and the iteration *t* *($P_{i(j)}(k,t)$)* can be computed by the equation shown in the figure below:

$$P_{i(j)}(k,\ t) = \frac{[P_{i(j)}(t)]^{\alpha}[\mu_{i(j)}(t)]^{\beta}}{\sum_{L_{i(j)}}[P_{i(j)}(t)]^{\alpha}[\mu_{i(j)}(t)]^{\beta}}$$

Figure 1.8, equation of **Ant Colony Optimization (ACO)** extracted from *Flood Hydrograph Prediction Using Machine Learning Methods.*

Where $P_{i(j)}(t)$ is the level of pheromones connected to option, $L_{i(j)}$, at the iteration t; $\mu_{i(j)} = L_{i(j)}/c_{i(j)}$ is a heuristic that prefers options with lower local costs; $c_{i(j)}$ is the set cost which has connecting options with $L_{i(j)}$. α and β are the exponents that determines the relative standing of pheromone and local heuristic factors respectively. The trail of the pheromone is updated using the equation in the figure below:

$$P_{i(j)}(t+1) = \delta P_{i(j)}(t) + \Delta P_{i(j)}$$

Figure 1.9, equation of **Ant Colony Optimization (ACO)** extracted from *Flood Hydrograph Prediction Using Machine Learning Methods.*

where $\Delta P_{i(j)}$ is the modification in pheromone concentration connected to option $L_{i(j)}$ and δ which is the pheromone persistence coefficient ($\delta < 1$). Using δ has many benefits like expanding the search space, preventing premature convergence, and avoiding expensive solutions.

Artificial Neural Network (ANN)

Artificial Neural Network or ANN refers to computing systems where the ideology was reference from the analogy of the biological neural networks. ANN are also commonly known as Neural nets, connectionist systems, parallel distributed processing systems, and neural nets. (Dongare & Kharde & Kachare, 2012). When the term “neural” is mentioned, the human brain comes into mind. Artificial Neural Networks refers to a biologically inspired sub-section of artificial intelligence modelled mainly after the human brain. An ANN is commonly a computational network based off the biological neural networks which gives shape to the system of the human brain. Like a brain, artificial neural networks also consist of neurons which links to each other in many layers in the network while inside the human brain, it is known as neurons, in ANN terms they are identified as nodes.

ANN in the field of artificial intelligence will undergo and attempt to mimic the network of neurons of a human brain therefore allowing machines to understand and make decisions like a normal human would behave. It is designed by programming the computers to behave like interconnect brain cells. It is estimate that there are about 1000 billion neurons located inside the brain of a human where each neurons have a linking point estimated in the range of about 1,000 to 100,000. Inside the human brain, data is stored this way to be distributed and each piece of data can be extracted, when necessary, from the memory parallelly. Therefore, it can be said that the human brain is structured out of extraordinary parallel processes. ANN takes this concept as a reference during development. (javaTpoint, n.d)

Within many of the machine learning methods, ANN perhaps are some of the most widely used methods in the modelling of flood predictions and any other natural disasters predictions. The ANN method excels at analysing nonlinear and multivariate data, as well as performing universal modelling. (Ngo et al, 2018). Despite these benefits, the use of ANNs in (Geographical Information System) GIS-based modelling of flash flood susceptibility remains limited. Besides that, previous studies that used ANN in spatial modelling of natural hazards frequently relied on gradient-based algorithms with backpropagation as a standard method for training the models. This traditional approach updates the weights of an ANN model during the training phase to minimise prediction errors.

Examples of work done with Artificial Neural Network

The chromosome’s suitability, F(C_i), in a pool of genes which contains a number of chromosomes is found by the equation in the figure below:

$$F(C_i) = \frac{f(C_i)}{\sum_{i=1}^{N} f(C_i)}$$

Figure 1.10, the equation used in the **Genetic Algorithm (GA)** algorithm, extracted from *Flood Hydrograph Prediction Using Machine Learning Methods.*

N is the total chromosomes amount while C_i is the ith chromosome. $f(C_i)$ stands for the functional value of ith chromosome and $F(C_i)$ represents the fitness value of ith chromosome.

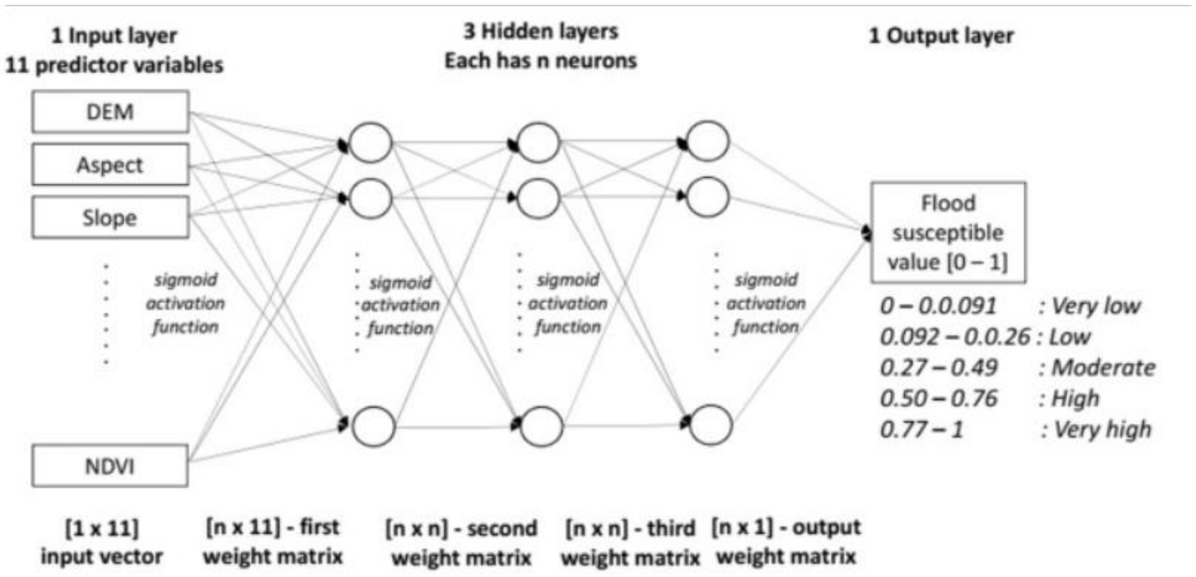


Figure 1.11, **Deep learning Neural Network (DNN)** structure extracted from *A novel hybrid approach based on a swarm intelligence optimized extreme learning machine for flash flood susceptibility mapping*

The DNN structure has 4 weight matrixes which have the respective sizes of n x 11, n x n, n x n, and n x 1. According to Bui et al. (2019), these weights are trained by the proposed optimization algorithms

$$f(x)= \frac{1}{1+e^{-x}}$$

$$RMSE = \sqrt{\frac{1}{n}\sum_{1}^{n}(predicted_i - observed_i)}$$

Figure 1.12, equation for the **Deep learning Neural Network (DNN)** structure extracted from *A novel hybrid approach based on a swarm intelligence optimized extreme learning machine for flash flood susceptibility mapping*

Flood prediction

The increase in frequency of floods cause by natural phenomenon is of great threat to mankind in terms of safety, economy and living surroundings. This is due to the increase in global warming which has led to the rise in frequency of the occurrence in urban areas across the globe. Therefore, early flood warnings systems are reliable countermeasures against the calamity of flood disasters which can minimize the incoming damage. With the use of machine learning technologies, many applications out there are very reliable due to the flexibility and scalability in terms of information extraction from complicated data structures and giving greater performance and more cost-effective solutions (Chang, et al. 2019).

Informative systems are important in geoscience and environmental control which is by providing integrated multi-control platforms which combines the data from visualizations, management, analysis, information on communication scale, and management. In recent times, an interest for designing early flood warning systems for storms that are considered extreme in urban areas are growing in popularity. This shows the increase in flood warning system developments due to the advancements in many weather forecasts, the effective use of high accuracy satellite data, techniques in the AI department, and advance technology in communications and information sharing. Early flood warning systems are regarded as the most used tool when it is related to flood predictions in urban areas with the high possibilities for emergency measures to counter the problem. It is also regarded as a cost-efficient option for safety, damage minimalization and possibly prevention (Yang, et al. 2018).

2.3 Study Area

According to the website *slbckerala.com* (n.d), the state of Kerala is situated at the Southwestern coast of India, almost inside the equatorial region. It is also known by another name; God’s Own country as how the locals call it. It is a tropical paradise where the Arabian sea is situated on the West and the Western Ghats on the East. This state is situated between the North latitudes of 8o18’ and 12o48’ and the East longitudes of 74o52’ and 77o22’. The area has a square kilometre of 38,863 square kilometres. It has a coast of 590 km, and the width of the state varies between 11 – 121 kilometres. The climate of Kerala is under the category of the wet tropical climate. The maximum temperature is 34 C, and the minimum is 22 C in the coastal and midland areas whereas 29 C is the maximum and 18 C is the minimum in the highland areas.

Kerala has 44 rivers. 41 of them are West flowing and 3 of them are East flowing. All of them in which are originated from the Western Ghats. The West flow reaches the Arabian Sea and the East flow reaches inside the Bay of Bengal. The big rivers are the Valapattanam river (110 km), the Chaliar river (69 km), Kadalundipuzha (130 km), Bharathapuzha (209 km), Chalakudy river (130 km), Periyar river (244 km), Pamba river (176 km), Achancoil (128 km), and Kalladayar river (121 km).

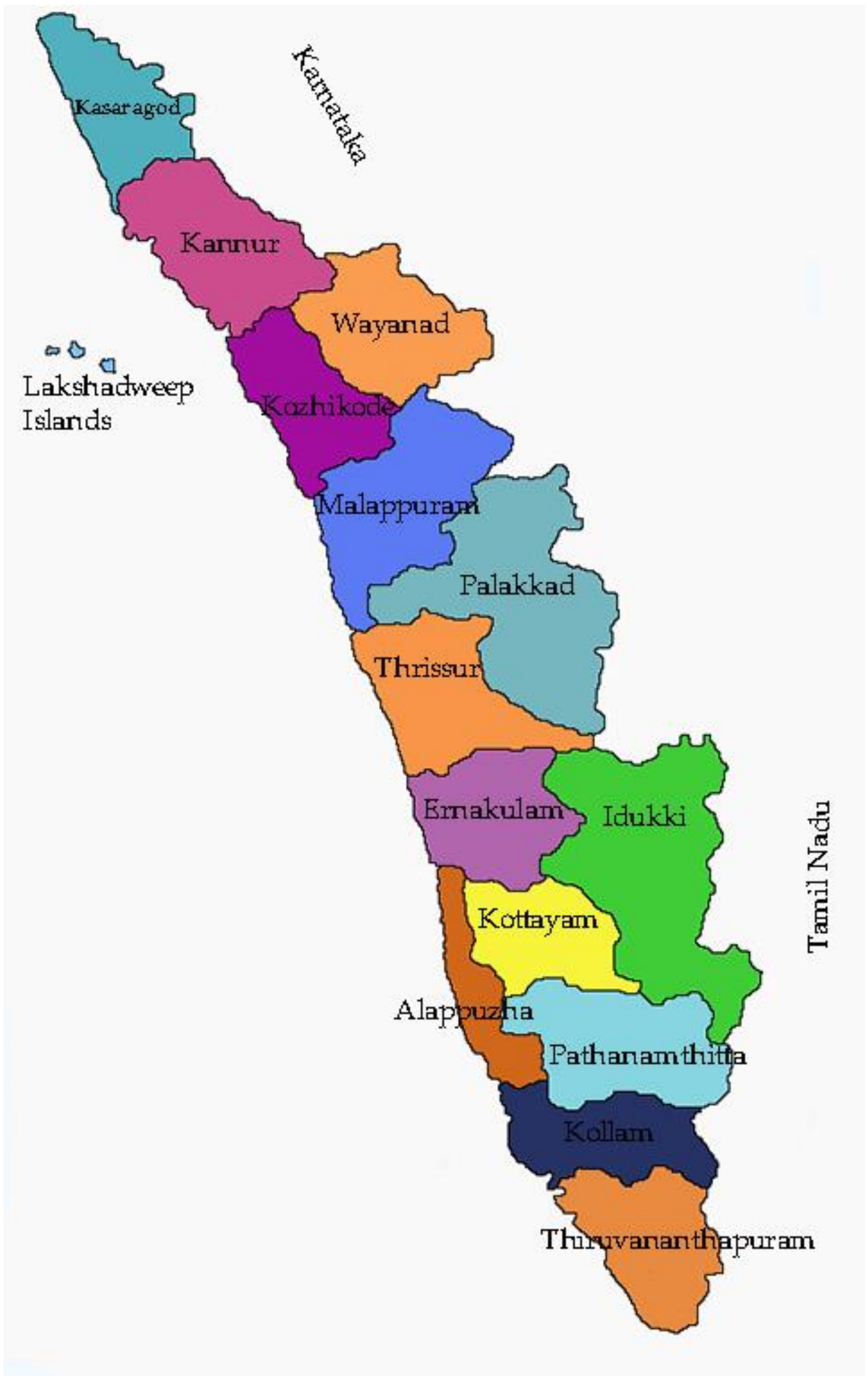


Figure 1.13, The map of the Kerala state in India extracted from *State Level Bankers Committee, Kerala (slbckerala.com)*

2.4 Flood Prediction Algorithms

Machine Learning (ML) forecasts floods by using mathematical expressions and algorithms. Furthermore, the models are data-driven, which means they are based on previous data history and lead to the improvement of prediction systems through the display of accurate results in a cost-effective manner. The first step was to from before the the dataset by looking for any null values and filling them in. Second, the descriptive data in the dataset is converted to numerical format because ML models cannot work directly with categorical data. After all, the data has been encrypted to quantitative form, the dataset is split into two parts: train dataset (70%) but also test dataset (30%). This step is a requirement for the verification of the model that was trained by comparing trained model results to test data results. (Kadiyala & Woo, 2022).

2.4.1 Support Vector Machine

Support Vector Machine or SVM for short, is a supervised machine learning model that can be used for both classification and regression. It transforms data using the kernel trick, and then formation an optimal bound (hyperplane) to differentiate between the possible outputs. SVM classifies information into various support vectors, then needs to draw an optimal hyperplane between the vectors while preserving the length between the hyperplane as well as the vectors as small as possible.

2.4.2 Logistic regression

Logistic Regression is a type of supervised learning model used to solve classification problems. It is applied when the output must fall within the 0 or 1 range, Yes or No, True or False, High or Low. This algorithm is based on the following equation:

$$\log\left[\frac{y}{1-y}\right] = b_0 + b_1 x_{1+} b_2 x_2 + \cdots + b_n x_n$$

Figure 2, equation extracted from *Flood Prediction and Analysis on the Relevance of Features using Explainable Artificial Intelligence*

2.4.3 Decision Tree

Decision trees are supervised learning models that can be used in alike regression and classification problems. It is made up of root, internal, and leaf nodes. This same decision tree works besides making the choice question the root node and extending the tree based on the question until the lowest of entropy is reached. The entropy formula is as follows:

$$\sum_{i=1}^{k} P(value_i)\ log_2(P(value_i))$$

Figure 2.1, formula extracted from *Flood Prediction and Analysis on the Relevance of Features using Explainable Artificial Intelligence*

where k is the number of elements in the dataset and P is just the probability of such an element The decision tree provided an accuracy of 75%, which is relatively low, implying that this is not the best algorithm for flood prediction.

2.4.4 K-Nearest Neighbour

The K-Nearest Neighbour(KNN) algorithm is one of the simplest and most easy to use machine learning algorithm based on supervised learning techniques. The K-NN algorithm presumes similarity among the new case/data and existing cases and places the new case in the category that is most like the existing categories (javaTpoint, n.d). It stores all the data available and classifies a new data point based off the similarities. The KNN algorithm can be used in both regression and classification problems but most of the time, it is used in the classification problems.

2.4.5 Random Forest

The Random Forest (RF) algorithm is a well-known machine learning algorithm which is categorised under the supervised learning technique. Similar to the KNN algorithm, it can be used for both regression and classification problems (javaTpoint, n.d). The concept of the RF algorithm is that it ensembles learning where it combines multiple classifiers to solve a complex problem and further improve the performance of the model. Instead of using just one decision tree, the RF algorithm takes the prediction from several trees and makes a final prediction output based on the majority of the votes of predictions.

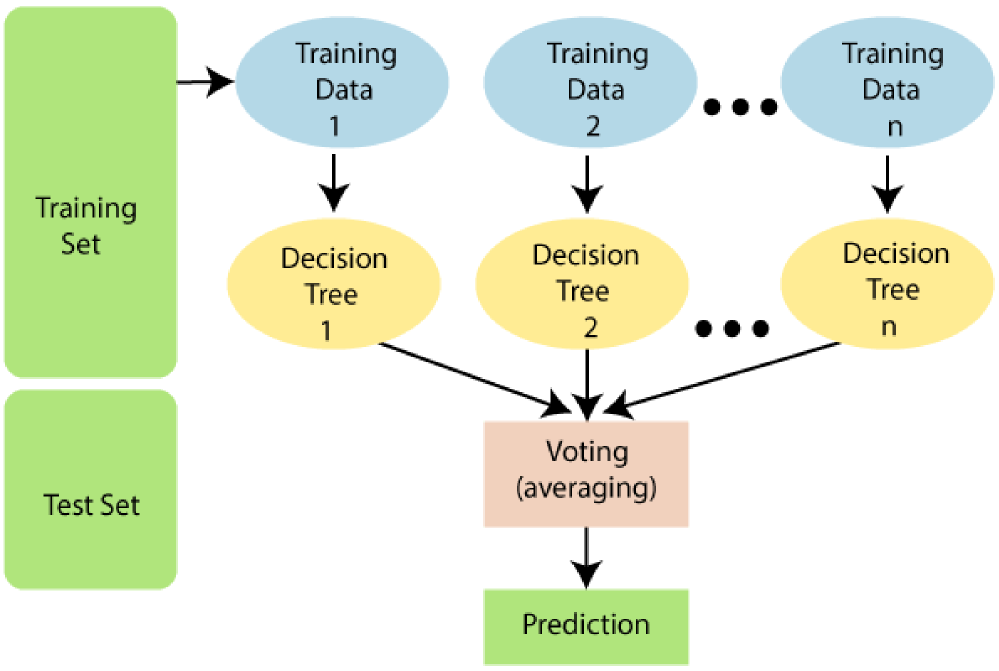


Figure 2.2, Diagram of a working RF algorithm extracted from *JavaTPoint: Random Forest Algorithm*

2.5 Particle Swarm Optimization (PSO)

In the passing of previous years, many optimizations techniques which are inspired by nature has come into surface in the education industry of artificial intelligence where it includes the Genetic Algorithms (GA), Ant Colony Optimization (ACO), Bacterial Foraging Optimization (BFO), Artificial Bee Colony (ABC), Grasshopper Optimization Algorithm (GOA), and Particle Swarm Optimization (PSO). Among these nature-inspired optimization techniques, the particle swarm optimization technique has been particularly promising with its output results.

According to Fadlallah et al. (2021), it is shown that it can produce effective optimization results for solving high constrained non-linear and non-convex optimization problems. Particle Swarm Optimization is an algorithm that was proposed by Kennedy and Eberhart back in 1995. As discussed in the original paper, sociobiologists have a belief where, be it a school of fish or possibly a flock of birds that moves in a group "can benefit from the experience of one another". Simply to break it down, one bird is flying to scout for food, all other birds in the flock can benefit from the discoveries they make and help the entire flock get the best hunt for food supply. While the movement of the birds can be simulated, it can also be considered that each bird is there to assist in finding the best solution in a high-dimensional solution space, and the best solution found by the flock is the best solution in the space (Tam, 2021). This solution is considered a heuristic solution because real global optimal solution can never be proven however it is often found that PSO produces results that are close to the global optimal result.

2.6 Feedforward Neural Network

A handful of problems in our daily lives usually involves some sort of intelligence pattern recognition, and object detections which proves to be quite the challenge in automating (Kurama, 2022). However, it is observed that it seems to be performed rather well and naturally by animals and children. Some of the examples would be a dog being able to recognise its owner over a stranger or a growing child being able to differentiate an apple and an orange. Such capabilities lie in the biological neural networks that are present in the human's nervous system. An "artificial neural network" on the other hand, is a computational system that mimics, or tries to mimic, the neural connections of the human nervous system. Initially, neural network was used only for classification problems, however with the increase in computational power in present days, there are now much more powerful architectures capable of solving more complicated problems. One of them is identified to be the Feedforward neural network.

The Feedforward neural network is known to be one of the first and most successful machine learning algorithm. They are also identified as deep networks, multi-layer perceptron (MLP), or just neural networks. The following is the structure of a feedforward neural network:

- **Input layer**
  - A layer consisting of neurons which receives inputs and pass them to the other layers. Number of neurons should be the same as the attributes of the existing dataset.
- **Output layer**
  - Output layer is the predicted results or features and is determined by the existing type of model.
- **Hidden layer**
  - Located between the input and output layer. Contains a vast number of neurons which is used for transformation of inputs before passing them onwards. If the network is trained. The weights become more predict-accurate.
- **Neuron weights**
  - Weights is also known as strength or amplitude of the connection between two neurons. These are often changed to small random values, like values in the range of 0 to 1.

2.7 Related works

There are a lot of works that does flood prediction via swarm intelligence. For instants, the work done and documented in the article *A novel hybrid approach based on a swarm intelligence optimized extreme learning machine for flash flood susceptibility mapping*. In this article, Bui et al. (2019) proposed the computing approach of an integration which is an Extreme Learning Machine (ELM) and the Particle Swarm Optimization (PSO) which forms the name PSO-ELM which will be used for spatial prediction of flash floods. The purpose of the ELM is to generate an initial state flood model while the PSO will be used as the optimization algorithm for the model. The Extreme Machine Learning (ELM) according to Bui et al. (2019), is a one-of-a-kind algorithm whereby it is used for a single-hidden layer feedforward neural network also known for short, SLFNs. Comparing the approach to a conventional least square support vector machine (LS-SVM), it is claimed to be able to generate respectable generalisation abilities at a significantly faster pace of learning and the proximal support vector machine (PSVM).

Another work example that also used swarm intelligence would be the article titled *Flood spatial prediction modelling using a hybrid of meta-optimization and support vector regression modelling*. The susceptibility of flood prediction is the initial important step in developing food mitigation plans to reduce the damage that could be dealt by flood disasters. The study in this article describes the optimizations of the support regression vectors (SVR) with the use of meta-optimization algorithm such as the Grasshopper Optimization Algorithm (GOA) to model the flood in urban flood areas. The grasshopper optimization algorithm was proposed in this article where according to Panahi et al. (2021) the SVR-GOA model was the best performance. In this work article, PSO was also used in the flood modelling which is the SVR-PSO model which came second in term of performance. The Grasshopper Optimization Algorithm is classified as a swarm intelligence algorithm which had its idea inspired from the swarm behaviours of grasshoppers as stated in the algorithm's name. The search process in GOA is the division of exploration and exploitation of the grasshoppers' repulsion and attraction forces where one grasshopper represents a specific set of values that will be used in the optimized areas.

Next, the article titled *A hybridized model based on neural network and swarm intelligence-grey wolf algorithm for spatial prediction of urban flood-inundation* discusses about the Grey Wolf Optimizer (GWO) algorithm. According to Darabi et al. (2021), the GWO algorithm is a novel bionic that is populated based as well as a heuristic optimization algorithm which takes the shape of a pyramid whereby the most dominant wolf in the remaining wolves which are in descendant of importance in low situations. In a mathematically perspective, this algorithm copies the nature of social leadership hierarchy in hunting natures while referring to the grey wolves working as a team to hunt down their prey. Three stages of prediction exist in the GWO algorithm: hunting, encircle then attack. The GWO algorithm offers faster evolutionary programming and faster convergence than the swarm intelligence algorithm due to its minimal number of parameters and ease of implementation.

As a follow up in this field of study, another method of flood prediction was done using the Ant Colony Optimization (ACO) algorithm. In the article titled *Flood Hydrograph Prediction Using Machine Learning Methods*, studies of how the algorithm was implemented in flood forecasting. According to Tayfur et al. (2018), this algorithm was developed by Dorigo who questioned how ants determine the fastest route from their colony to a food supply. A chemical tracer known as pheromone was discovered to be used by ants to communicate despite their total blindness. On their travels, ants leave behind pheromones. A path with the highest concentration of pheromone is hence likely to be chosen by ants. This implies that if there are both long and short paths between the nest and the food supply, the ants will finally use the shortest possible path.

Besides that, the very same article *Flood Hydrograph Prediction Using Machine Learning Methods,* presented the Genetic Algorithm (GA) which is under Artificial Neural Network (ANN). ANN is also used in flood predictions can be used alongside swarm optimization algorithms like the PSO and GWO algorithm. According to Tayfur et al. (2018) the Genetic Algorithm algorithm was developed by John H. Holland, author of the book *Adaptation In Natural and Artificial System*. With the concept where it is survival of the fittest, it uses chromosomes where each are of many genes per say. For every gene, it signifies a specific decision variable, and each chromosome signifies a possible solution that is considered optimal. It is noted that each gene is formed from a string consisting of a series of digits which is between 0 and 1. (refer to figure 1.1)

Other ANN methods like Deep learning Neural Network (DNN) is also used in flood predictions. Extracted from the article *A novel hybrid approach based on a swarm intelligence optimized extreme learning machine for flash flood susceptibility mapping*, Bui et al. (2019) mentions that DNN is from the family tree of ANN where it is specified to have several hidden layers. It is a feed-forward network that is typically trained using the back-propagation method, however when hidden layer numbers increase and learning rates in hidden layers vary, the network becomes more challenging to train. There are no general conditions on the number of hidden layers and the number of neurons are there in the layers since according to Bui et al. (2019), the decision is subjected to the problem’s complexity and condition of the existing dataset.

3.0 Comparison of Related Works

With in-depth analysis of the related works found for flood prediction with swarm intelligence and artificial neural networks, comparison can be made with the algorithms used when predicting floods. With the usage of the hybrid PSO-ELM model based on the findings of Bui et al. (2019), an accuracy of 93.34% and 90.05% were concluded respectively for the training and the validation of datasets. The results are clear on the graphs of the proposed model where the performance was measured for both training and validation of the datasets. Research on an ANN model was also done by Bui et al. (2019), which is the MLP-ANN and SVM that has a validation accuracy of 88.01% and 87.24% respectively. It was recorded that the PSO-ELM has successfully outperformed the other machine learning models.

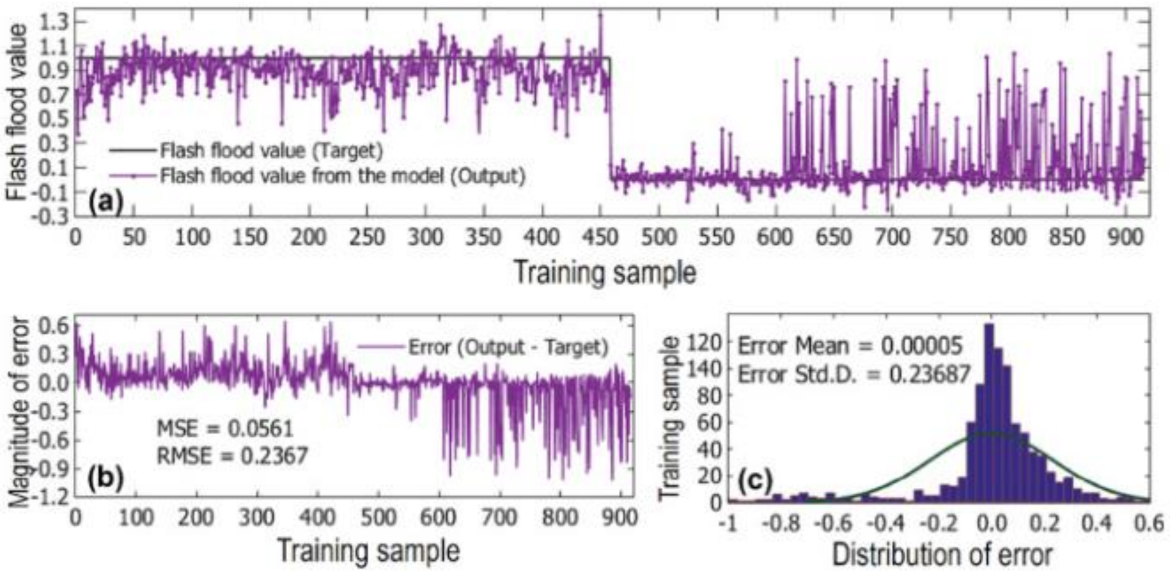


Figure 3.1, Training phase of the PSO-ELM model extracted from *A novel hybrid approach based on a swarm intelligence optimized extreme learning machine for flash flood susceptibility mapping.*

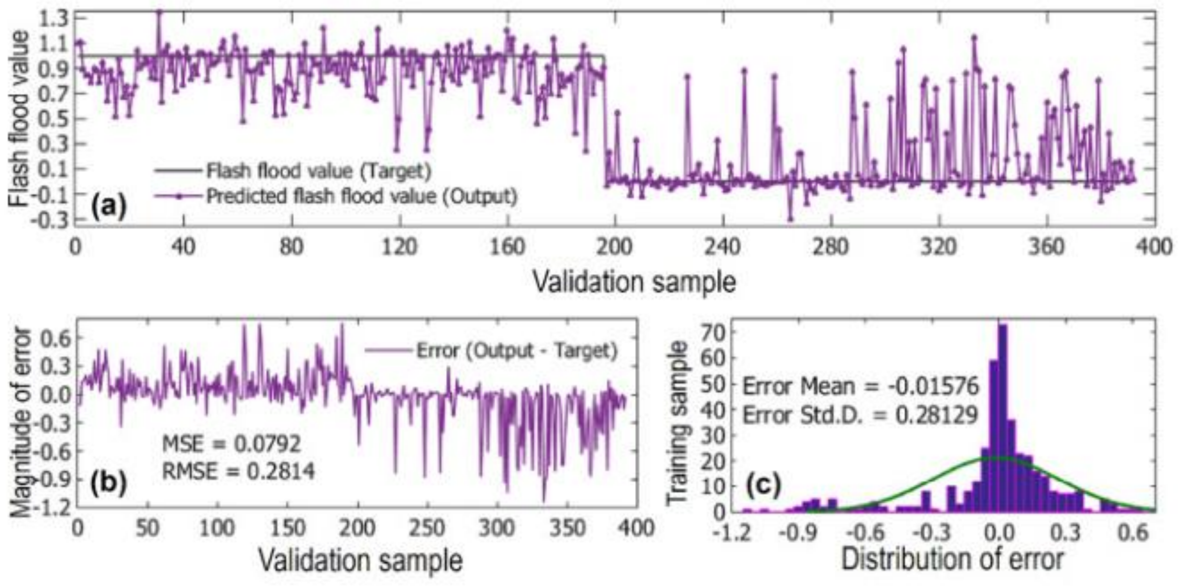
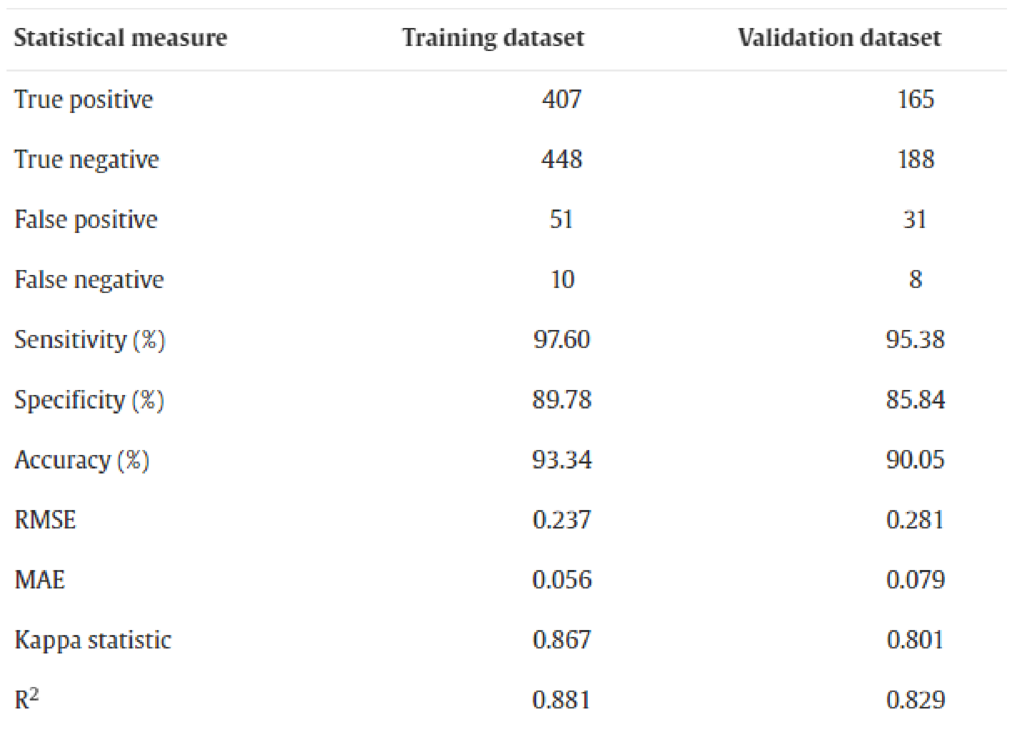
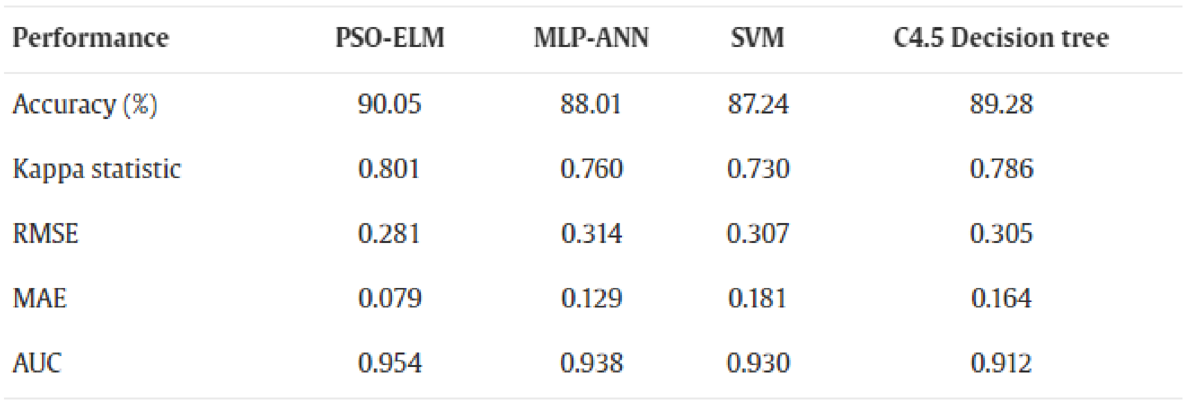


Figure 3.2, Performance of the PSO-ELM model extracted from *A novel hybrid approach based on a swarm intelligence optimized extreme learning machine for flash flood susceptibility mapping.*



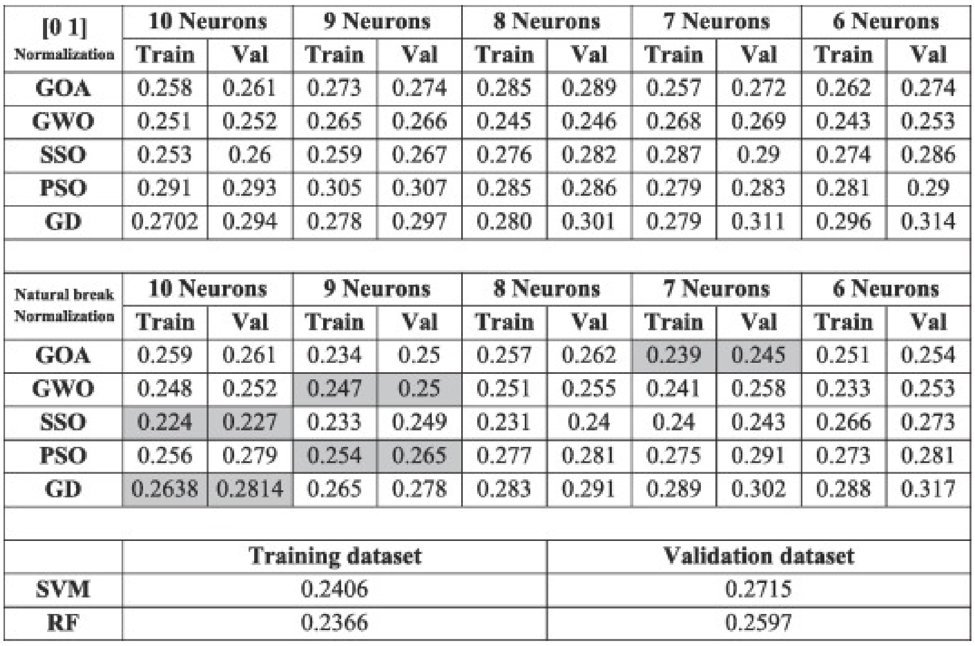
Statistical measure	Training dataset	Validation dataset
True positive	407	165
True negative	448	188
False positive	51	31
False negative	10	8
Sensitivity (%)	97.60	95.38
Specificity (%)	89.78	85.84
Accuracy (%)	93.34	90.05
RMSE	0.237	0.281
MAE	0.056	0.079
Kappa statistic	0.867	0.801
R^2	0.881	0.829



Performance	PSO-ELM	MLP-ANN	SVM	C4.5 Decision tree
Accuracy (%)	90.05	88.01	87.24	89.28
Kappa statistic	0.801	0.760	0.730	0.786
RMSE	0.281	0.314	0.307	0.305
MAE	0.079	0.129	0.181	0.164
AUC	0.954	0.938	0.930	0.912

Figure 3.2, Statistics tables of the performance of the PSO-ELM model extracted from *A novel hybrid approach based on a swarm intelligence optimized extreme learning machine for flash flood susceptibility mapping.*

In addition, the work done by Bui et al. (2020) shows a variation of RMSEs with GOA, SSO, and GWO. It was observed and recorded that GOA and GWO has similar searching patterns which jumps from the beginning to the 500th interation almost instantly and then gradually decreases until it reaches the maximum repetition. Its strategy was to begin the exploring stage which aims to search for potential areas which then is followed up by an exploitation search to find the approximate best global value. As for SSO, it took aggregated signals from the other searching agents, in this case spider, and is seen to have little jumps in the constant combinations that changes with the search.

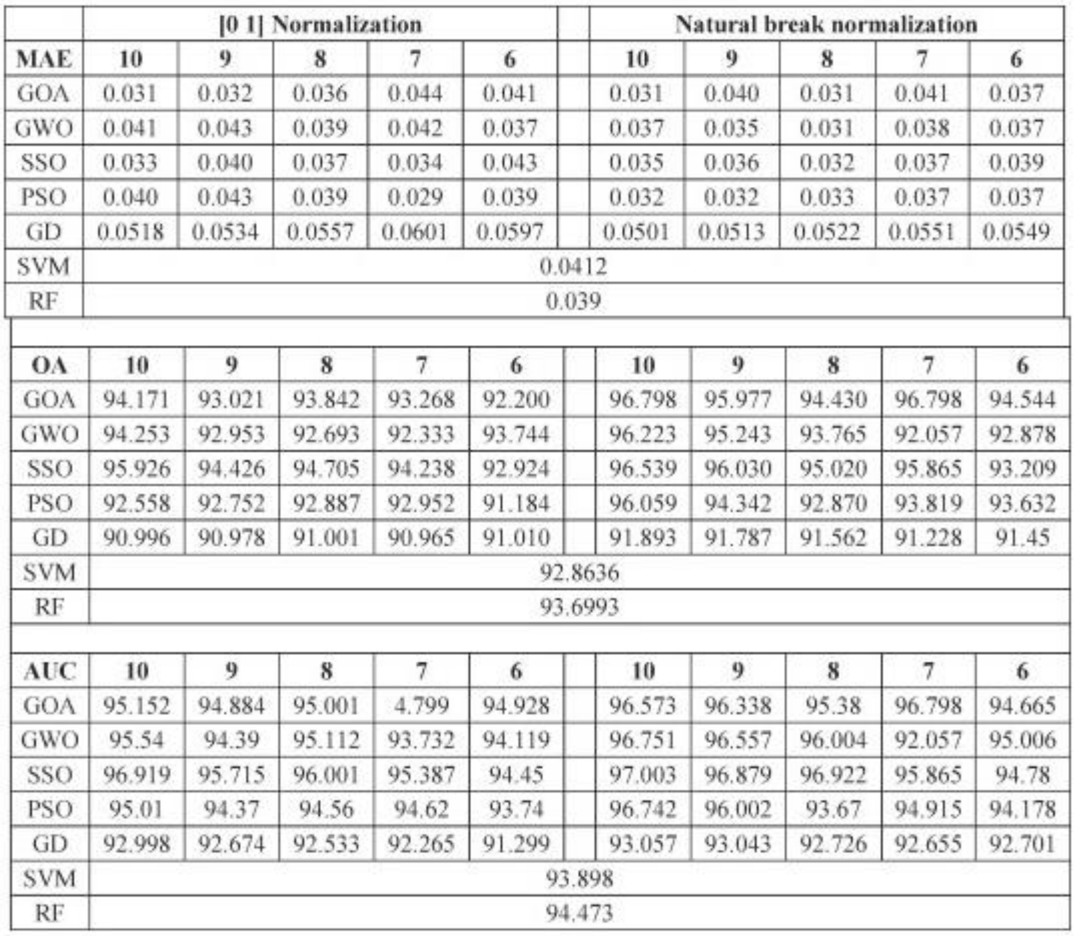


[0 1] Normalization	10 Neurons		9 Neurons		8 Neurons		7 Neurons		6 Neurons	
	Train	Val	Train	Val	Train	Val	Train	Val	Train	Val
GOA	0.258	0.261	0.273	0.274	0.285	0.289	0.257	0.272	0.262	0.274
GWO	0.251	0.252	0.265	0.266	0.245	0.246	0.268	0.269	0.243	0.253
SSO	0.253	0.26	0.259	0.267	0.276	0.282	0.287	0.29	0.274	0.286
PSO	0.291	0.293	0.305	0.307	0.285	0.286	0.279	0.283	0.281	0.29
GD	0.2702	0.294	0.278	0.297	0.280	0.301	0.279	0.311	0.296	0.314

Natural break Normalization	10 Neurons		9 Neurons		8 Neurons		7 Neurons		6 Neurons	
	Train	Val	Train	Val	Train	Val	Train	Val	Train	Val
GOA	0.259	0.261	0.234	0.25	0.257	0.262	0.239	0.245	0.251	0.254
GWO	0.248	0.252	0.247	0.25	0.251	0.255	0.241	0.258	0.233	0.253
SSO	0.224	0.227	0.233	0.249	0.231	0.24	0.24	0.243	0.266	0.273
PSO	0.256	0.279	0.254	0.265	0.277	0.281	0.275	0.291	0.273	0.281
GD	0.2638	0.2814	0.265	0.278	0.283	0.291	0.289	0.302	0.288	0.317

	Training dataset	Validation dataset
SVM	0.2406	0.2715
RF	0.2366	0.2597

Figure 3.3, RMSEs training(Train) and validation(Val) datasets of two normalization methods extracted from *Verification of novel integrations of swarm intelligence algorithms into deep learning neural network for flood susceptibility mapping.*



	[0 1] Normalization					Natural break normalization				
MAE	**10**	**9**	**8**	**7**	**6**	**10**	**9**	**8**	**7**	**6**
GOA	0.031	0.032	0.036	0.044	0.041	0.031	0.040	0.031	0.041	0.037
GWO	0.041	0.043	0.039	0.042	0.037	0.037	0.035	0.031	0.038	0.037
SSO	0.033	0.040	0.037	0.034	0.043	0.035	0.036	0.032	0.037	0.039
PSO	0.040	0.043	0.039	0.029	0.039	0.032	0.032	0.033	0.037	0.037
GD	0.0518	0.0534	0.0557	0.0601	0.0597	0.0501	0.0513	0.0522	0.0551	0.0549
SVM	0.0412									
RF	0.039									
OA	**10**	**9**	**8**	**7**	**6**	**10**	**9**	**8**	**7**	**6**
GOA	94.171	93.021	93.842	93.268	92.200	96.798	95.977	94.430	96.798	94.544
GWO	94.253	92.953	92.693	92.333	93.744	96.223	95.243	93.765	92.057	92.878
SSO	95.926	94.426	94.705	94.238	92.924	96.539	96.030	95.020	95.865	93.209
PSO	92.558	92.752	92.887	92.952	91.184	96.059	94.342	92.870	93.819	93.632
GD	90.996	90.978	91.001	90.965	91.010	91.893	91.787	91.562	91.228	91.45
SVM	92.8636									
RF	93.6993									
AUC	**10**	**9**	**8**	**7**	**6**	**10**	**9**	**8**	**7**	**6**
GOA	95.152	94.884	95.001	4.799	94.928	96.573	96.338	95.38	96.798	94.665
GWO	95.54	94.39	95.112	93.732	94.119	96.751	96.557	96.004	92.057	95.006
SSO	96.919	95.715	96.001	95.387	94.45	97.003	96.879	96.922	95.865	94.78
PSO	95.01	94.37	94.56	94.62	93.74	96.742	96.002	93.67	94.915	94.178
GD	92.998	92.674	92.533	92.265	91.299	93.057	93.043	92.726	92.655	92.701
SVM	93.898									
RF	94.473									

Figure 3.4, Mean Absolute Error(MAE), Overall Accuracy(OA), and Area Under ROC(AUC) datasets of two normalization methods extracted from *Verification of novel integrations of swarm intelligence algorithms into deep learning neural network for flood susceptibility mapping.*

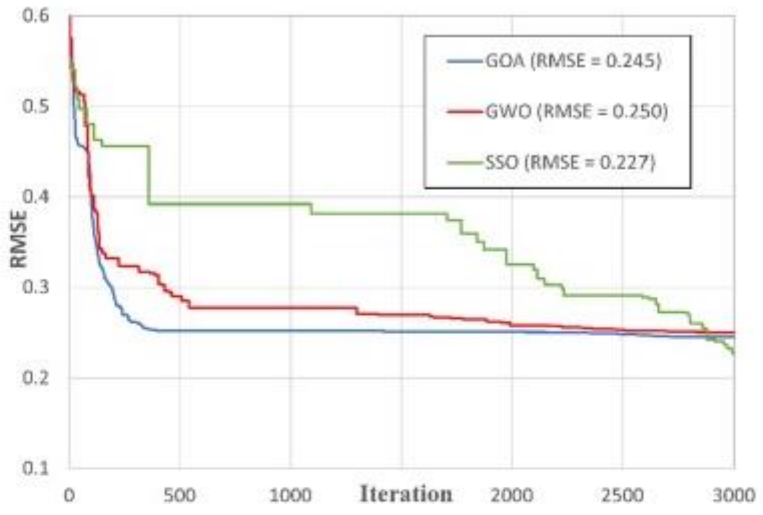


Figure 3.5, Variations of RMSEs extracted from *Verification of novel integrations of swarm intelligence algorithms into deep learning neural network for flood susceptibility mapping.*

In terms of the Root Mean Square Error (RMSE), Mean Absolute Error (MAE) and Area Under the Curve (AUC) done by Panahi et al. (2021), comparison of the SVR-GOA, SVR-PSO, and SVR model was done with these three results. The results were taken for the validation assessment of the hybrid model where both SVR-PSO and SVR-GOA outperforms the standalone SVR. However, when compared in terms of AUC, both the SVR-PSO and SVR-GOA has similar values of 0.959. Although there are slight difference in values when compared to the RMSE and the MSE results, according to Panahi et al. (2021), both PSO and GOA has successful improved the standalone SVR by an approximate 9.1% in the AUC.

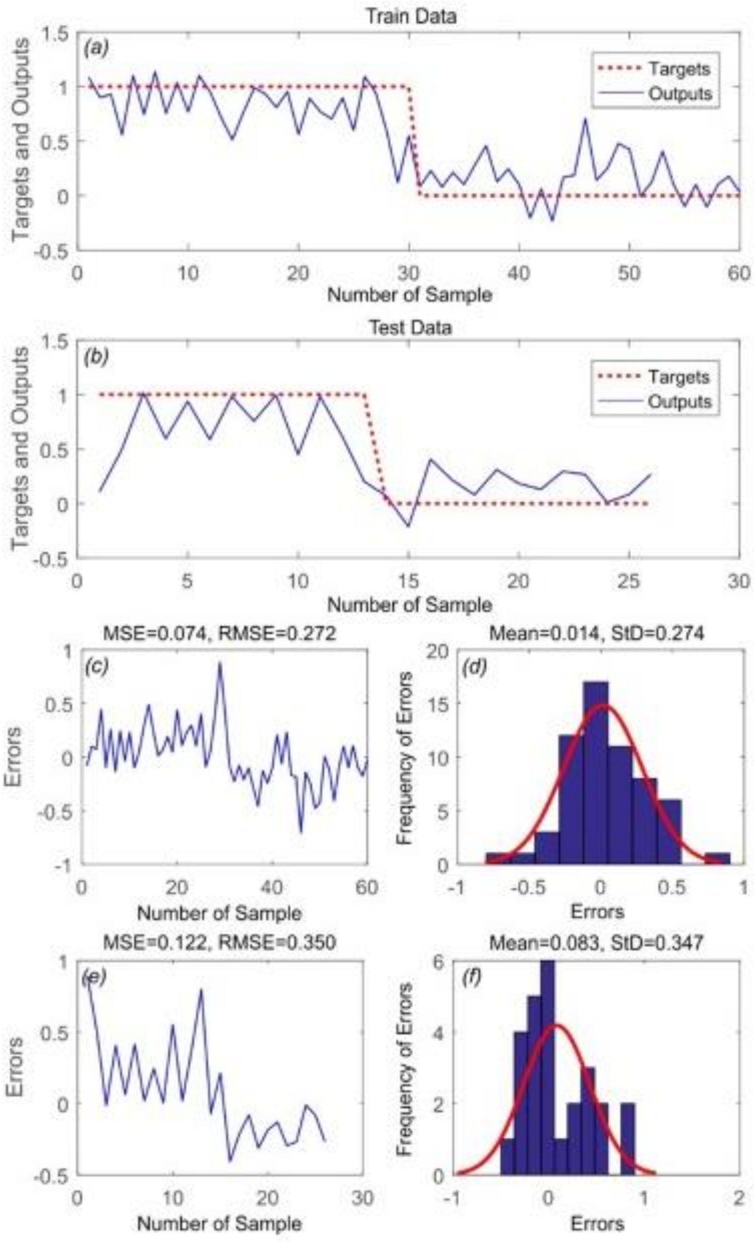


Figure 3.6, SVR model results. From top: SVR simulation vs target data for learning phase, SVR simulation vs target data of validation phase, MSE and RMSE values of learning data, frequency error of learning, MSE and RMSE values of validation data, and frequency error of validation data extracted from *Flood spatial prediction modelling using a hybrid of meta-optimizing and support vector regression modelling.*

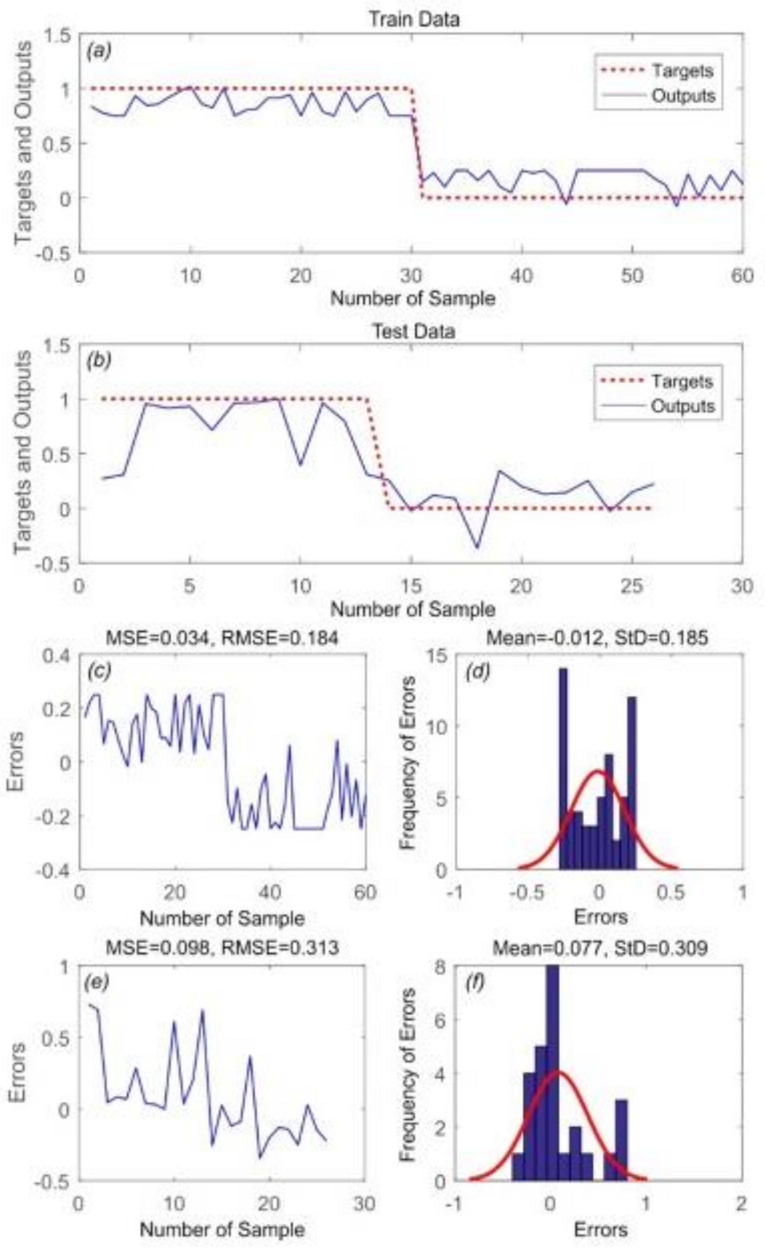


Figure 3.7, SVR-GOA model results. From top: SVR-GOA simulation vs target data for learning phase, SVR-GOA simulation vs target data of validation phase, MSE and RMSE values of learning data, frequency error of learning, MSE and RMSE values of validation data, and frequency error of validation data extracted from *Flood spatial prediction modelling using a hybrid of meta-optimizing and support vector regression modelling.*

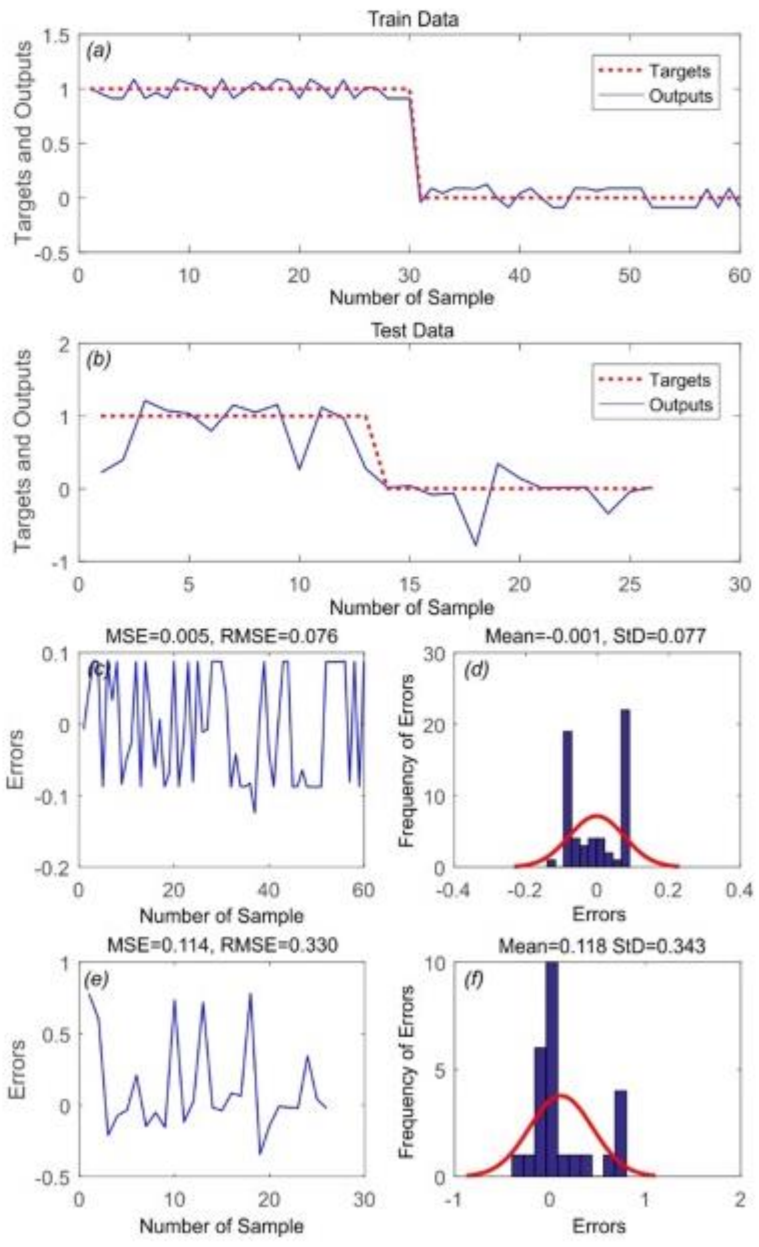


Figure 3.8, SVR-PSO model results. From top: SVR-PSO simulation vs target data for learning phase, SVR-PSO simulation vs target data of validation phase, MSE and RMSE values of learning data, frequency error of learning, MSE and RMSE values of validation data, and frequency error of validation data extracted from *Flood spatial prediction modelling using a hybrid of meta-optimizing and support vector regression modelling.*

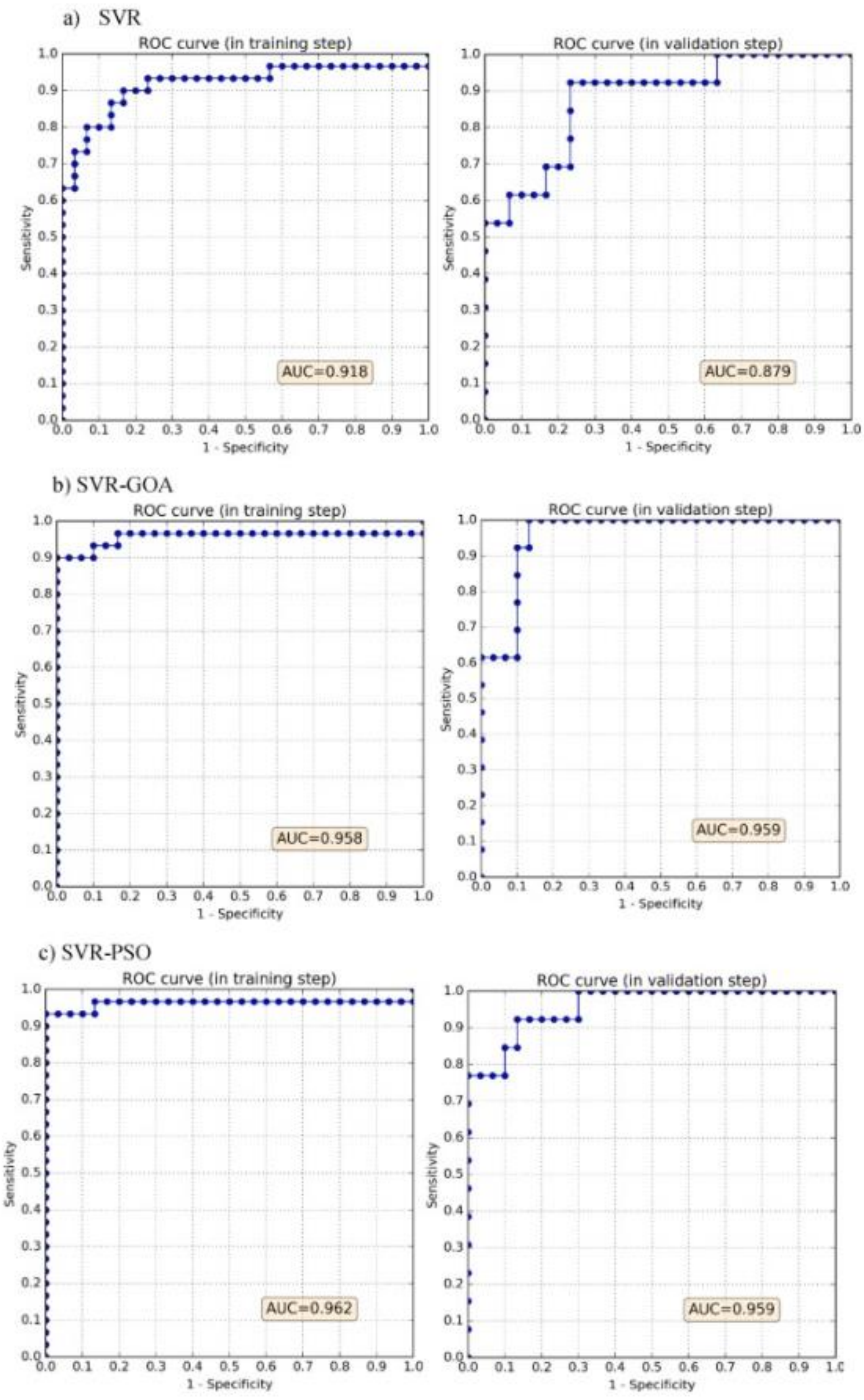
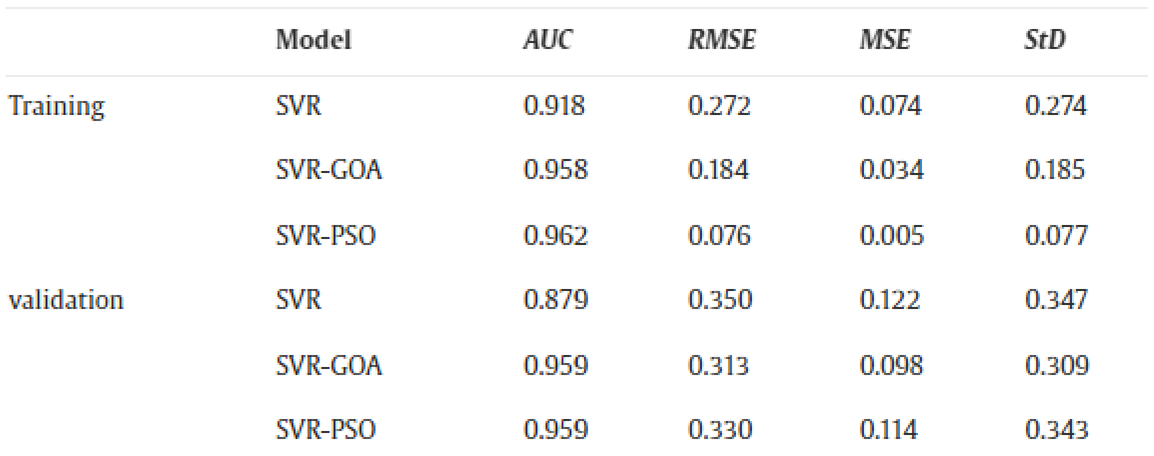


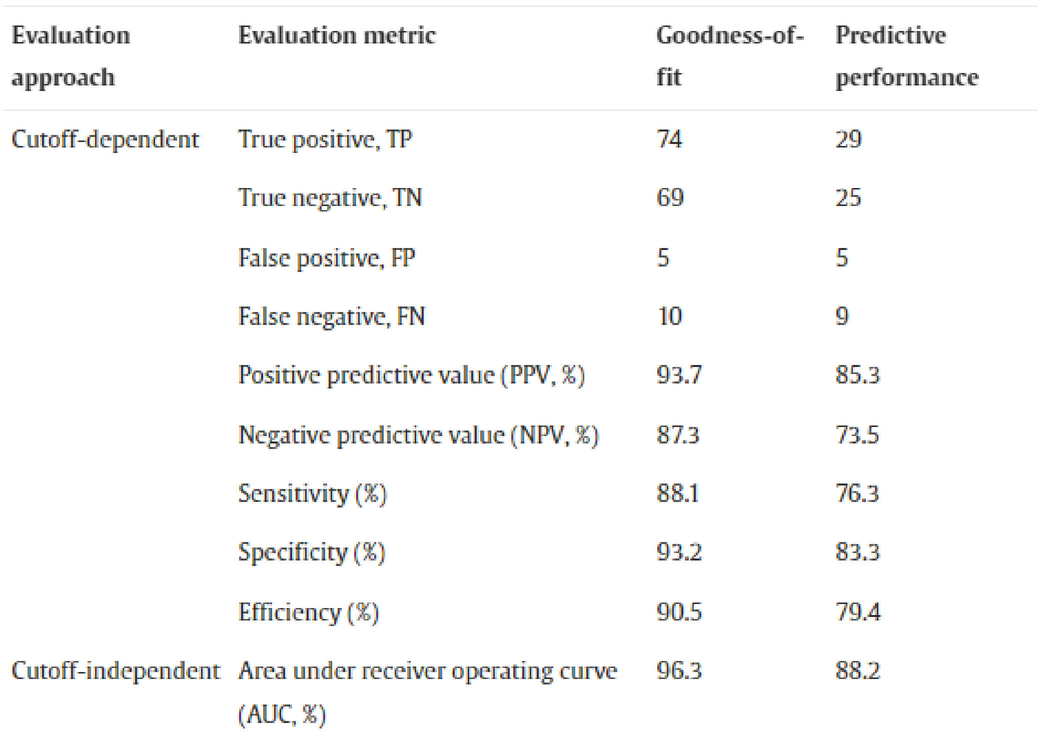
Figure 3.9, ROC curves of the SVR model, SCR-GOA model, and SVR-PSO model extracted from *Flood spatial prediction modelling using a hybrid of meta-optimizing and support vector regression modelling.*



	Model	*AUC*	*RMSE*	*MSE*	*StD*
Training	SVR	0.918	0.272	0.074	0.274
	SVR-GOA	0.958	0.184	0.034	0.185
	SVR-PSO	0.962	0.076	0.005	0.077
validation	SVR	0.879	0.350	0.122	0.347
	SVR-GOA	0.959	0.313	0.098	0.309
	SVR-PSO	0.959	0.330	0.114	0.343

Figure 3.10, Accuracy assessment of hybrid models extracted from *Flood spatial prediction modelling using a hybrid of meta-optimizing and support vector regression modelling.*

There are also other studies in which ANN was used in the predictions of flood and that is exactly what Darabi et al. (2021) did in their studies using hybridized models such as the ANN-SGW model. In terms of AUC, the ANN-SGW model has recorded a result of 0.963, the highest AUC value in the table above when comparing with the other AUC values. This model proposed by Darabi et al. (2021) also has an accuracy of 90.50% in the validation phase. The performance of the ANN-SGW model was compared with four other benchmark models which are the Random Forest (RF), Logistic Model Tree (LMT), Classification And Regression trees (CART), and the J48 Decision Tree (J48DT). The AUC of the models are as the following; RF model(AUC=0.881), J48DT model(AUC=0.697), CART model(AUC=0.698), and LMT model(AUC=0.756). Based of the studies, a conclusion was made where the ANN-SGW model has the highest accuracy followed by the RF, LMT, CART, and J48DT models respectively.



Evaluation approach	Evaluation metric	Goodness-of-fit	Predictive performance
Cutoff-dependent	True positive, TP	74	29
	True negative, TN	69	25
	False positive, FP	5	5
	False negative, FN	10	9
	Positive predictive value (PPV, %)	93.7	85.3
	Negative predictive value (NPV, %)	87.3	73.5
	Sensitivity (%)	88.1	76.3
	Specificity (%)	93.2	83.3
	Efficiency (%)	90.5	79.4
Cutoff-independent	Area under receiver operating curve (AUC, %)	96.3	88.2

Figure 3.11, Performance of the hybridized model, ANN-SGW in the training and validation steps extracted from *A hybridized model based on neural network and swarm intelligence-grey wolf algorithm for spatial prediction of urban flood-inundation*

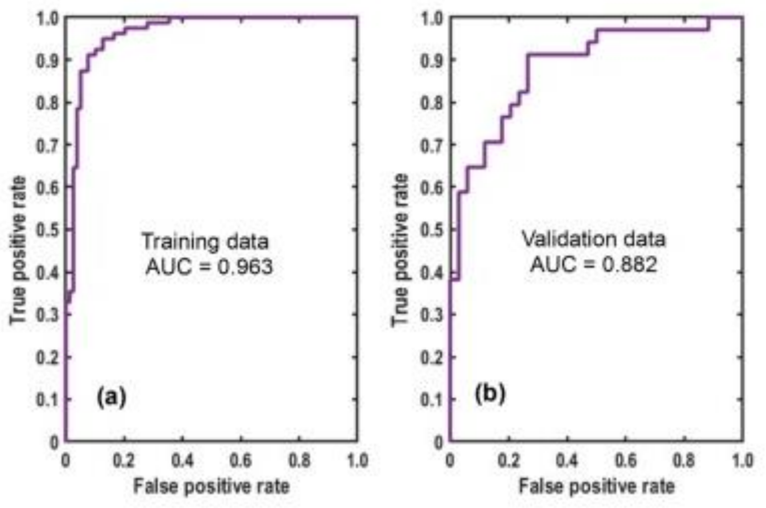
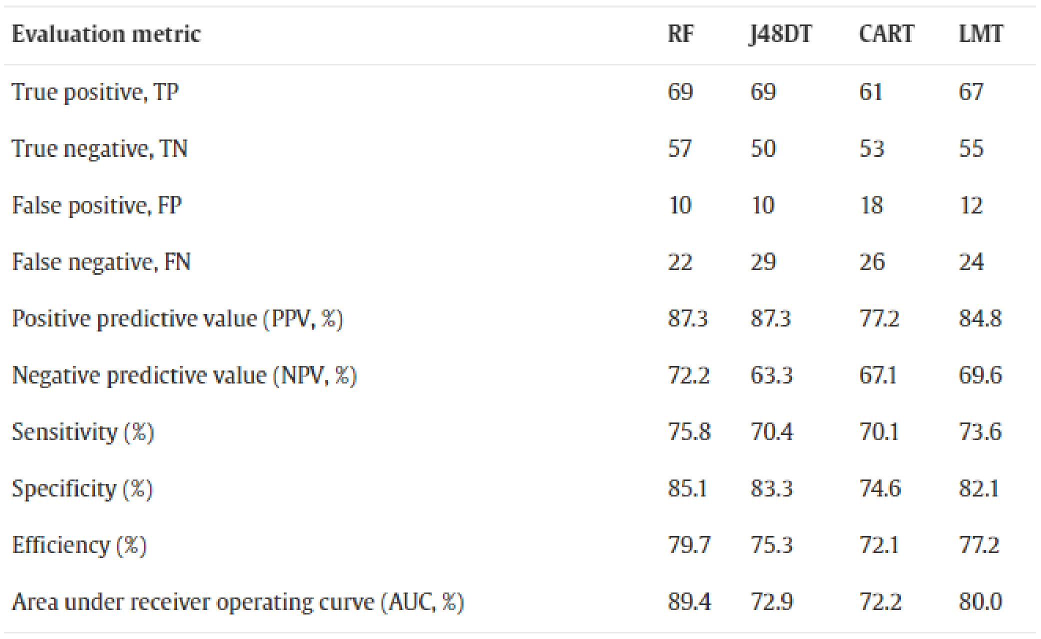
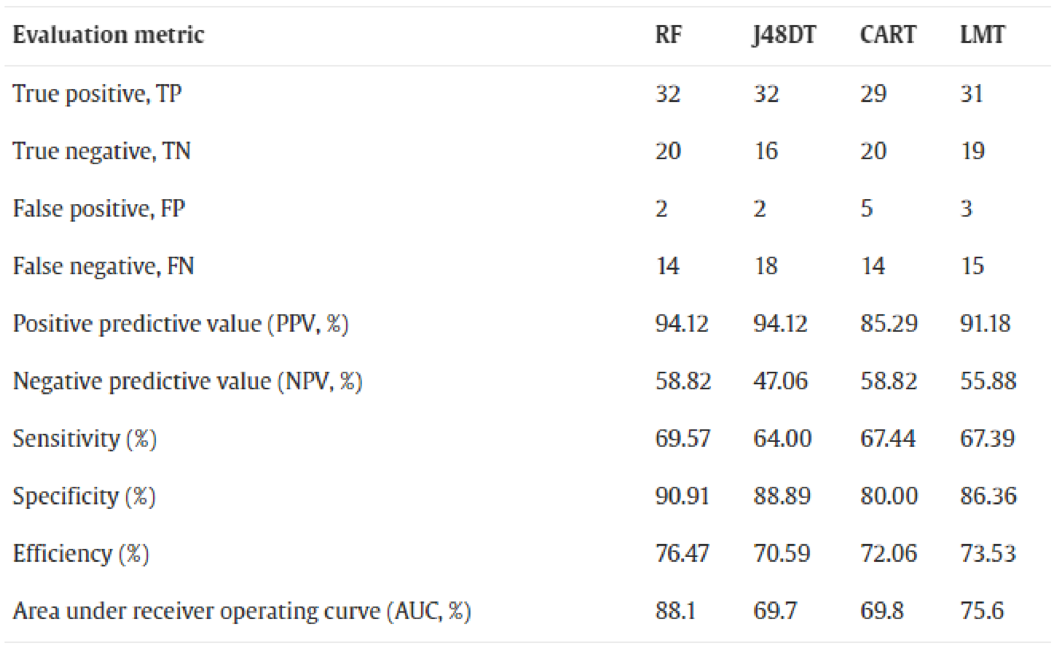


Figure 3.12, AUC of hybridized model ANN-SGW, left with the training dataset and right with the validation dataset extracted from *A hybridized model based on neural network and swarm intelligence-grey wolf algorithm for spatial prediction of urban flood-inundation*



Evaluation metric	RF	J48DT	CART	LMT
True positive, TP	69	69	61	67
True negative, TN	57	50	53	55
False positive, FP	10	10	18	12
False negative, FN	22	29	26	24
Positive predictive value (PPV, %)	87.3	87.3	77.2	84.8
Negative predictive value (NPV, %)	72.2	63.3	67.1	69.6
Sensitivity (%)	75.8	70.4	70.1	73.6
Specificity (%)	85.1	83.3	74.6	82.1
Efficiency (%)	79.7	75.3	72.1	77.2
Area under receiver operating curve (AUC, %)	89.4	72.9	72.2	80.0

Figure 3.13, Goodness of fit of benchmark models random forest (RF), J48 decision tree (J48DT), classification and regression trees (CART), logistic model tree (LMT) in training extracted from *A hybridized model based on neural network and swarm intelligence-grey wolf algorithm for spatial prediction of urban flood-inundation*



Evaluation metric	RF	J48DT	CART	LMT
True positive, TP	32	32	29	31
True negative, TN	20	16	20	19
False positive, FP	2	2	5	3
False negative, FN	14	18	14	15
Positive predictive value (PPV, %)	94.12	94.12	85.29	91.18
Negative predictive value (NPV, %)	58.82	47.06	58.82	55.88
Sensitivity (%)	69.57	64.00	67.44	67.39
Specificity (%)	90.91	88.89	80.00	86.36
Efficiency (%)	76.47	70.59	72.06	73.53
Area under receiver operating curve (AUC, %)	88.1	69.7	69.8	75.6

Figure 3.13, Predictive performance of benchmark models random forest (RF), J48 decision tree (J48DT), classification and regression trees (CART), logistic model tree (LMT) in training extracted from *A hybridized model based on neural network and swarm intelligence-grey wolf algorithm for spatial prediction of urban flood-inundation*

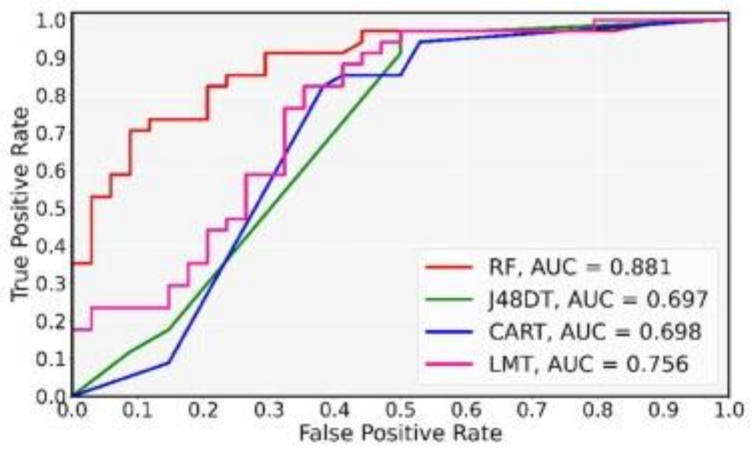


Figure 3.14, AUC of the benchmark models, random forest (RF), J48 decision tree (J48DT), classification and regression trees (CART), logistic model tree (LMT) in validation steep extracted from *A hybridized model based on neural network and swarm intelligence-grey wolf algorithm for spatial prediction of urban flood-inundation*

Moving back to RMSE and MAE, a study done by Tayfur et al. (2018), the results were recorded in m^3/s instead. According to Tayfur et al. (2018), the ANN, GA_RCM and ACO_RCM models produced and average result of 9 m^3/s in terms of RMSE and 7 m^3/s in terms of MAE while for the PSO_RCM is averaging about 13 m^3/s RMSE and 11 m^3/s MAE. While the RCM standalone made higher numbers of errors which are 13 m^3/s MAE and 17 m^3/s RMSE. This study shows that a hybridized model has proven a better optimization than the standalone model when it comes to the prediction results where the errors are reduced by a noticeable margin and therefore proving its effectiveness in prediction of flood.

These are some of the related works done in this field of prediction models and many have produced promising results in terms of the output data. The methodologies proposed are promising and shows qualities in terms of reliability with good reasonings. The same can be said with the proposed methods and algorithms in the studies in performing a prediction alongside comparisons of data as an evaluation step to the approaches that was proposed in these related works.

4.0 Proposed methodology

4.1 Source of dataset

The data of the flood predictions will be based on an annual rainfall dataset where the algorithm will refer the data from to make the flood predictions outputs. Such datasets can be found on website database that uploads these datasets from across the globe. The dataset will possibly be searched via the website Kaggle.com, GitHub, and the government's data portal which is Data.gov.my for the Malaysian government's data portal. However, due to the purpose of this capstone project where supervised learning method is the chosen method in the experiments, a supervised dataset is needed and therefore all obtained dataset of Malaysia was not used. A dataset of Kerala flood was chosen to be used due to it being a supervised dataset with confirmation of the flood areas that corresponds to the annual rainfall. The main goal of this project is to evaluate the accuracy of the algorithms.

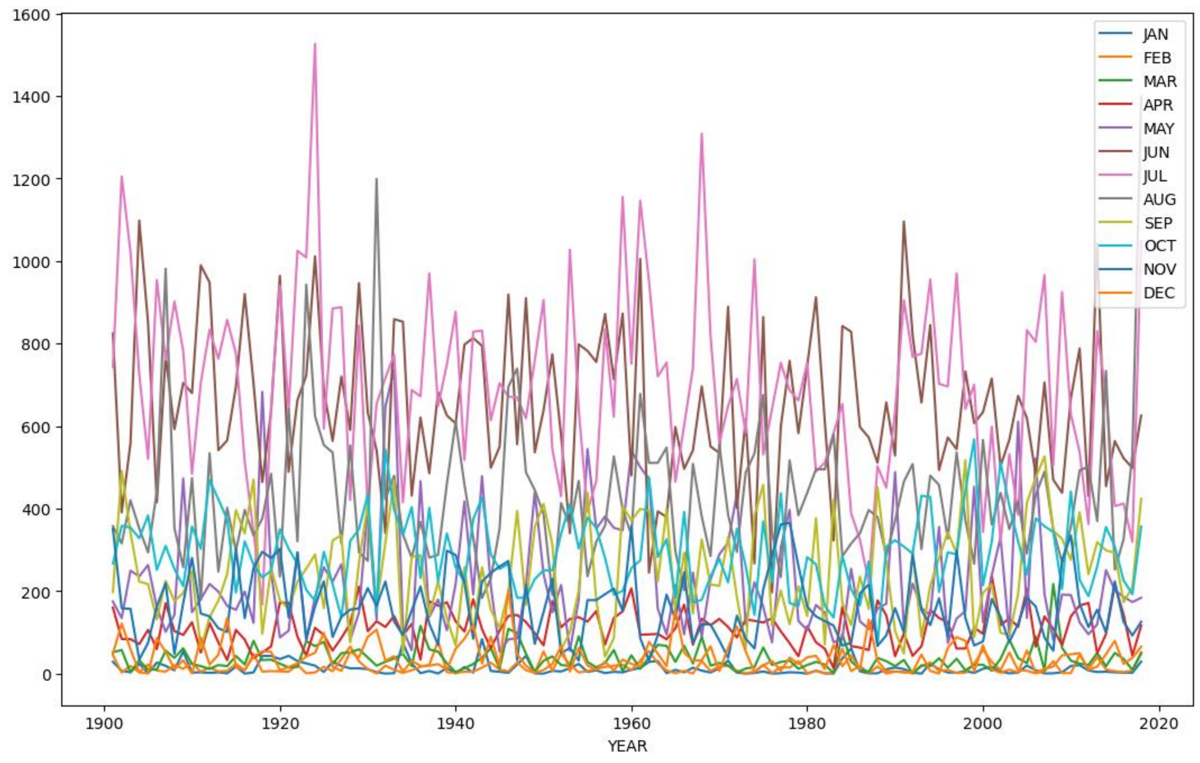


Figure 4.0, dataset of Kerala's monthly rainfall corresponding to the year

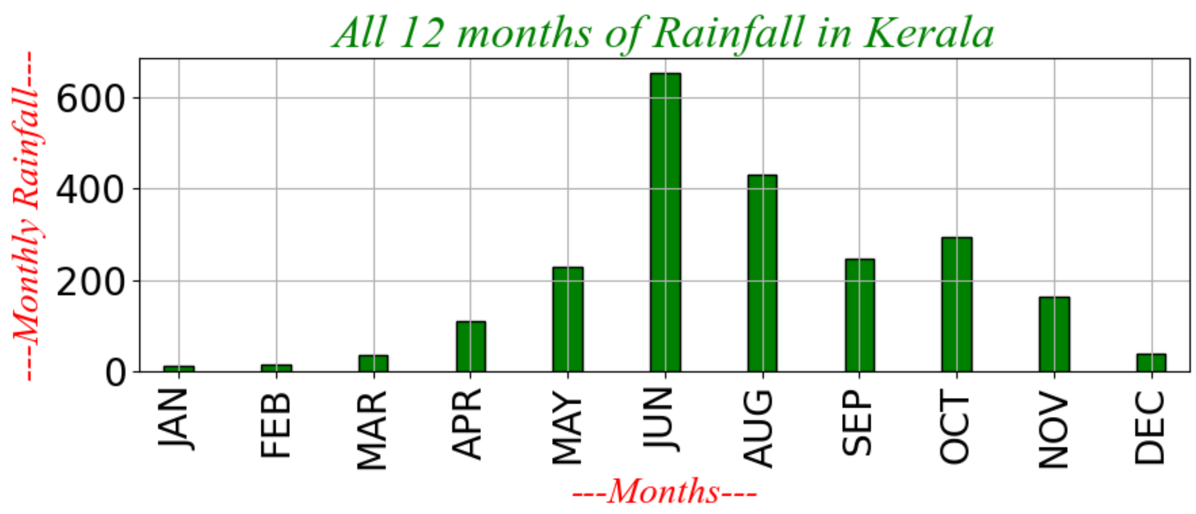
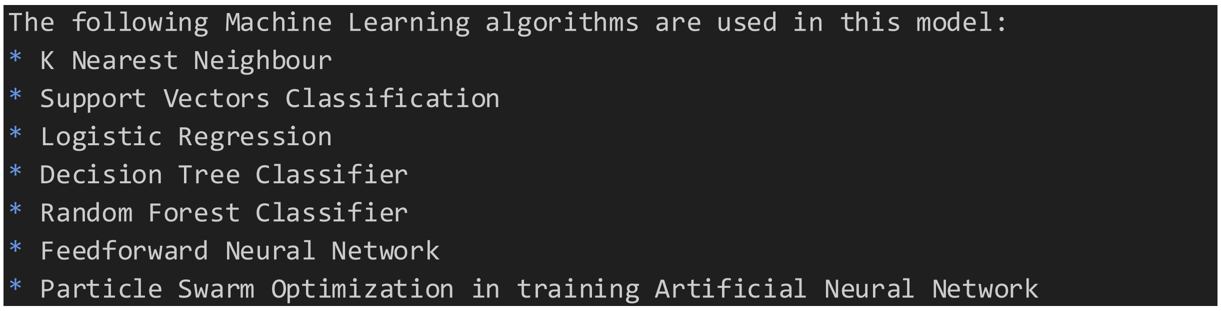


Figure 4.1, dataset of Kerala

4.2 Algorithms



```
The following Machine Learning algorithms are used in this model:
* K Nearest Neighbour
* Support Vectors Classification
* Logistic Regression
* Decision Tree Classifier
* Random Forest Classifier
* Feedforward Neural Network
* Particle Swarm Optimization in training Artificial Neural Network
```

The algorithms that were planned to be used are K Nearest Neighbour, Support Vectors Classification, Logistic Regression, Decision Tree Classifier, and Random Forest Classifier. These 5 machine learning algorithms will be used in one benchmark model. This benchmark model will be used to compare the accuracy with two other models. The two other models consist of a single algorithm. One being the Feedforward Neural Network and the other being the Particle Swarm Optimization in training Artificial Neural Network.

4.3 Technical plan

This section will highlight about the tools that will be used in carrying out the necessary tasks in this project such as what software will be used, which programming language, the programming language library, and possibly more.

4.4 Tools implementation

1) Software

The intended software that will be used to carry out the optimization and testing of the existing data would be Visual Studio Code through Anaconda. As it was more convenient and easier to use, and I personally find it easier to understand this two software. Besides that, they have several built-in extensions which would make the workload much easier because the extensions mostly provide pre-sets of code structures or easy imports of libraries.

2) Programming Language: Python
The Python language will be used in this optimization and testing process since most of the existing code since most of the existing code are coded using the python language with the python library. The reason why python was chosen is because python is much more straight forward to use therefore it is much simpler to analyse and modify if necessary. Most optimizations in the swarm intelligence pool also happens to be mostly python therefore making the wide variety of optimization methods available. Additionally, the usage of Jupyter Notebook was also used in this project as it makes the breakdown much clearer when debugging or when analysing.

3) Python library
The list of libraries planned to be used may change in the next phase of the project. The libraries that will be used are the NumPy library, Scikit-Learn library, the Pandas library, TensorFlow library, and Keras library are all used in this capstone project. These will be the core libraries that were planned to be used in our models.

4.5 Supervised Learning

In this capstone project, supervised learning, which is also labelled as supervised machine learning is used in the experiment on the dataset. The Kerala flood dataset is considered a labelled dataset since it already has pre-determined flood results and the year that flood did occur in the aforementioned urban area. The dataset also provides the monthly rainfall in all twelve months in a year and has a column for the annual rainfall total.

5.0 Particle Swarm Optimization in training Artificial Neural Network.

In this section, we will be discussing how the PSO-ANN model works and its functions in training the neural network which are applied with the Kerala data sets obtain from Kaggle.com.

Firstly, the Kerala dataset is loaded for the input data and the target data is loaded and split. 70% of the data will be used to train the ANN and 30% for the testing of the ANN.

Next, the neural network is then constructed. An example of the architecture which was used for the Kerala dataset is shown below.

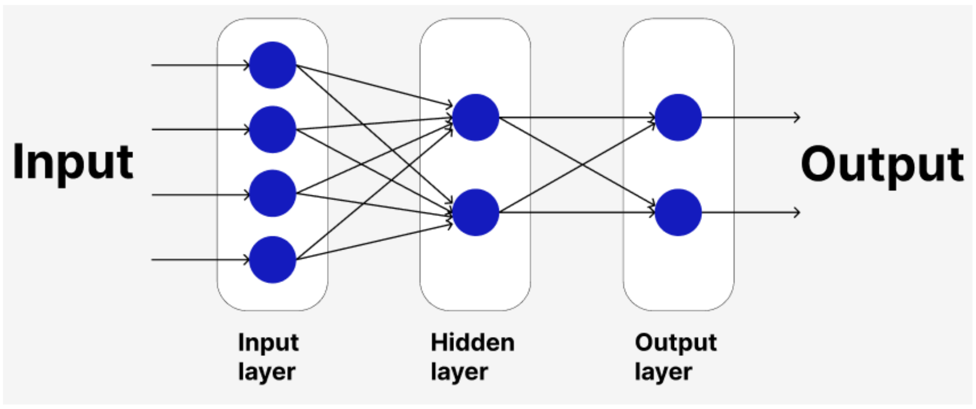
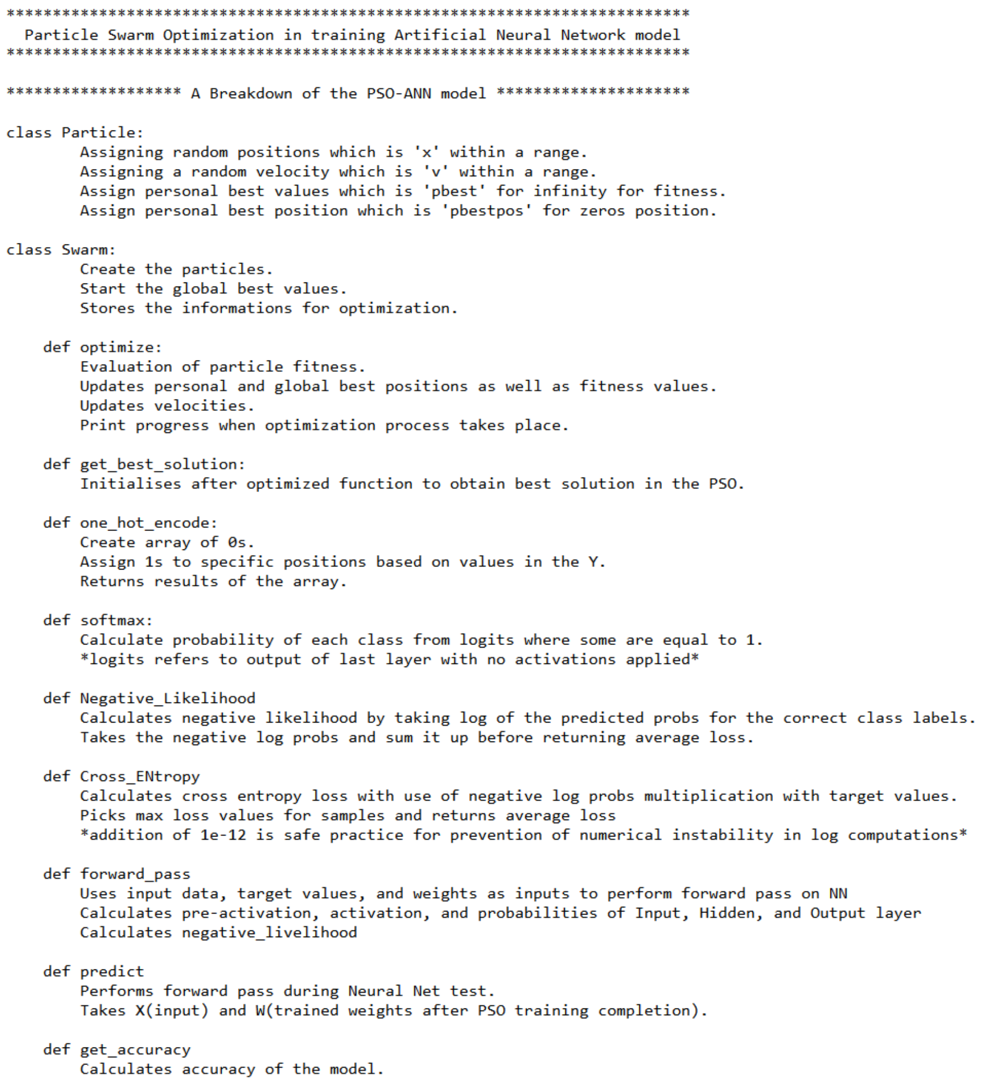


Figure 4.2, Architecture of the neural network drawn using figma.com

After constructing the neural network, the PSO algorithm is called where the optimization takes place and thus, train the ANN. A visual breakdown of the process is listed in the figure below for a clear picture of how the model functions and to provide a better understanding of the processes.



```
**************************************************************************
  Particle Swarm Optimization in training Artificial Neural Network model
**************************************************************************

****************** A Breakdown of the PSO-ANN model ********************

class Particle:
        Assigning random positions which is 'x' within a range.
        Assigning a random velocity which is 'v' within a range.
        Assign personal best values which is 'pbest' for infinity for fitness.
        Assign personal best position which is 'pbestpos' for zeros position.

class Swarm:
        Create the particles.
        Start the global best values.
        Stores the informations for optimization.

    def optimize:
        Evaluation of particle fitness.
        Updates personal and global best positions as well as fitness values.
        Updates velocities.
        Print progress when optimization process takes place.

    def get_best_solution:
        Initialises after optimized function to obtain best solution in the PSO.

    def one_hot_encode:
        Create array of 0s.
        Assign 1s to specific positions based on values in the Y.
        Returns results of the array.

    def softmax:
        Calculate probability of each class from logits where some are equal to 1.
        *logits refers to output of last layer with no activations applied*

    def Negative_Likelihood
        Calculates negative likelihood by taking log of the predicted probs for the correct class labels.
        Takes the negative log probs and sum it up before returning average loss.

    def Cross_ENtropy
        Calculates cross entropy loss with use of negative log probs multiplication with target values.
        Picks max loss values for samples and returns average loss
        *addition of 1e-12 is safe practice for prevention of numerical instability in log computations*

    def forward_pass
        Uses input data, target values, and weights as inputs to perform forward pass on NN
        Calculates pre-activation, activation, and probabilities of Input, Hidden, and Output layer
        Calculates negative_livelihood

    def predict
        Performs forward pass during Neural Net test.
        Takes X(input) and W(trained weights after PSO training completion).

    def get_accuracy
        Calculates accuracy of the model.
```

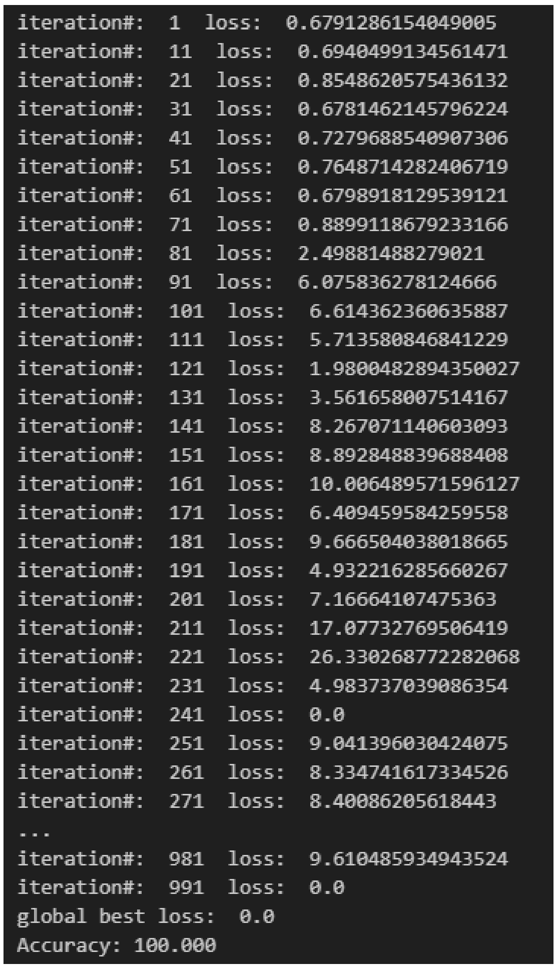
Figure 4.3, A visual breakdown of the PSO-ANN model done in a notepad.

The PSO algorithm then sets the number of particles in the model. In this case, 100 particles were set in the model. Then the number of dimensions for each of the particle in the PSO algorithm is calculated using the formula stated below.

$$\text{Num. Dim} = (\text{IL} * \text{HL}) + \text{HL} + (\text{HL} * \text{OL}) + \text{OL}$$

Figure 4.4, formula used in calculating number of dimensions for each particle in the PSO algorithm.

Then, the range of values for the weights, learning rates, and inertia weight are defined. As a follow up, the cognitive and social factors are also defined where the velocity is updated in via the PSO algorithm. The swarm then is initialising with the number of solutions, number of dimensions, weights range, learning rate range, inertia weight range, and cognitive in its parameters. Then the optimization takes place where it calls the forward pass function for the fitness function, following with it, the data inputs values, and the target values. A frequency of 10 for every iteration process is decided and a total of 1000 iterations is running in the optimization process.



```
iteration#:  1  loss:  0.6791286154049005
iteration#:  11  loss:  0.6940499134561471
iteration#:  21  loss:  0.8548620575436132
iteration#:  31  loss:  0.6781462145796224
iteration#:  41  loss:  0.7279688540907306
iteration#:  51  loss:  0.7648714282406719
iteration#:  61  loss:  0.6798918129539121
iteration#:  71  loss:  0.8899118679233166
iteration#:  81  loss:  2.49881488279021
iteration#:  91  loss:  6.075836278124666
iteration#:  101  loss:  6.614362360635887
iteration#:  111  loss:  5.713580846841229
iteration#:  121  loss:  1.9800482894350027
iteration#:  131  loss:  3.561658007514167
iteration#:  141  loss:  8.267071140603093
iteration#:  151  loss:  8.892848839688408
iteration#:  161  loss:  10.006489571596127
iteration#:  171  loss:  6.409459584259558
iteration#:  181  loss:  9.666504038018665
iteration#:  191  loss:  4.932216285660267
iteration#:  201  loss:  7.16664107475363
iteration#:  211  loss:  17.07732769506419
iteration#:  221  loss:  26.330268772282068
iteration#:  231  loss:  4.983737039086354
iteration#:  241  loss:  0.0
iteration#:  251  loss:  9.041396030424075
iteration#:  261  loss:  8.334741617334526
iteration#:  271  loss:  8.40086205618443
...
iteration#:  981  loss:  9.610485934943524
iteration#:  991  loss:  0.0
global best loss:  0.0
Accuracy: 100.000
```

Figure 4.5, Frequency of the iteration process and the total iterations running during the optimization process.

The best solution, which was found by the PSO algorithm, is than obtained and is then used to make predictions with the usage of the input data. It then calculates the accuracy of the predicted results with the target values and is then multiplied by 100 to obtain a percentage accuracy result.

The Root Mean Square Error (RMSE) results is determined by the maximum and minimum errors. In other words, it is very sensitive to both the max and min errors. This allows it to become an accuracy effectiveness tool for the evaluation of how accurate a prediction result can be. The RMSE value that leans closer to 0 indicates a higher accuracy effectiveness of the prediction result.

$$RMSE = \sqrt{\sum_{i=1}^{n} \frac{(Q_0 - Q_c)^2}{n}}$$

Figure 4.6, formula of the RMSE

For the calculation of the Mean Square Error (MSE) value. The formula stated in figure 5 is used but without the square root in the formula. The MSE is calculated using the squared difference average from both target values (Q0) and the predicted values (Qc). The RMSE is then obtained by just square rooting the MSE.

6.0 Experimental Results

6.1 Experimental Dataset

A dataset of Kerala flood is obtained from Kaggle which will be used in the experimental process. The Kerala dataset is a supervised dataset where it is already pre-determined whether the flood has already occurred it its respective year.

The Kerala dataset consists of 118 rows and 16 columns. 12 of the columns are the months in a year where monthly rainfall data are recorded for that year with a follow up total of rainfall in the year which is recorded under the column with the label Annual Rainfall. The following column after consisting of the label whether a flood has occurred or not in that year in the form of YES and NO. This column data determines that this dataset is suitable for the supervised learning method in this project.

6.2 Experiments carried out for the FFNN model.

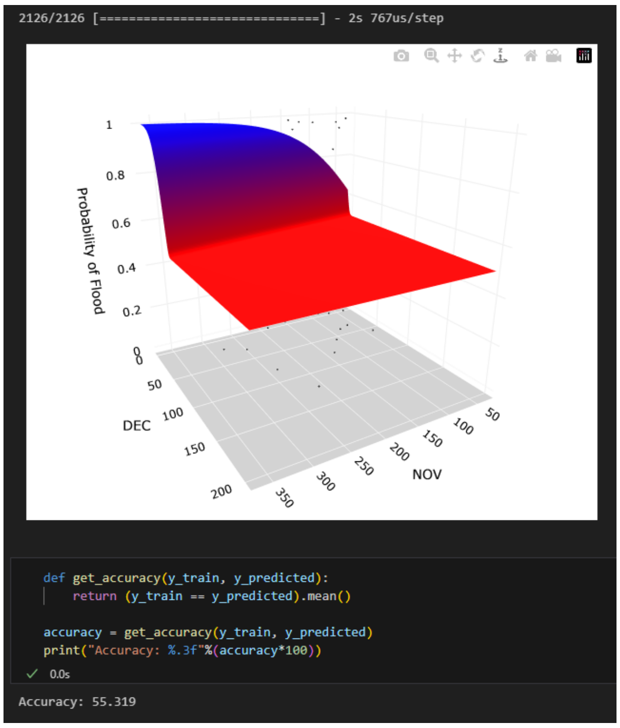
In these experimental attempts, the Kerala dataset will be used as the input to several models whereby the dataset will be split into test data and train data. The architectures of the neural network used in the Feedforward Neural Network (FFNN) is 1-2-1 and 2-2-1, being Input layer, Hidden layer, and Output layer respectively.

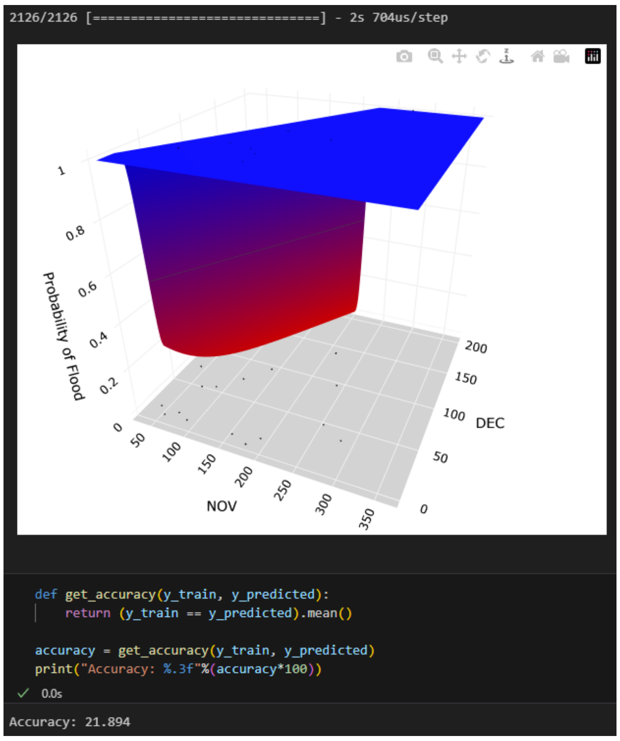
In this experimental phase, the focus is to evaluate the accuracy of model itself. By taking 1 to 2 inputs and then applying the kernels (also known as weights) and biases before passing the results through the activation functions. At the very end, we will get an output which is the prediction that consists of complex set of transformation that has been optimized with training. The neural network is trained by fitting the custom curve through the training data that is led by loss minimization and through the parameter (which consists of the kernels and biases) optimization. However, the main goal is to evaluate the accuracy of the model and therefore additionally we must calculate the accuracy based on the predicted and actual labels.

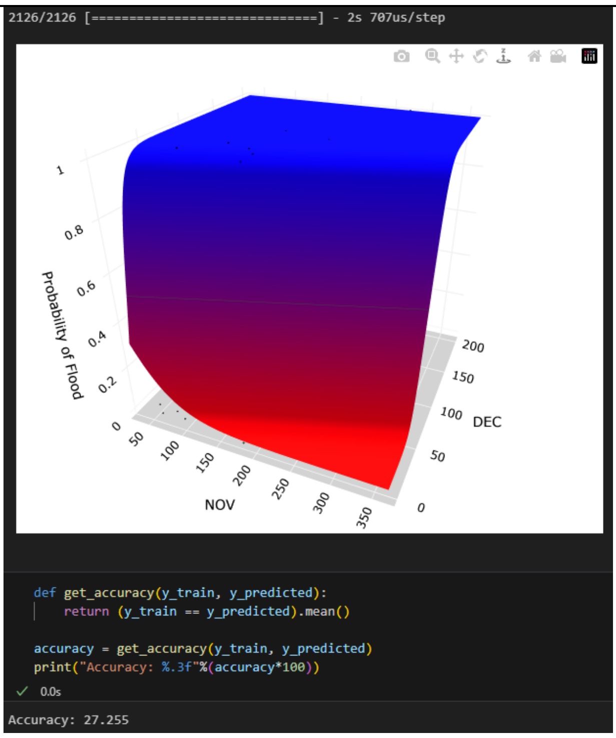
It was found that the accuracy varies by a huge margin (for example, the accuracy can vary from 0% to 55%) therefore the model will run a total of 10 times for observation of how high the accuracy can reach to. The following table is the results of the experimental phase for the FFNN model running 10 times to find the highest possible accuracy of what the FFNN model can reach through observation of the accuracy results.

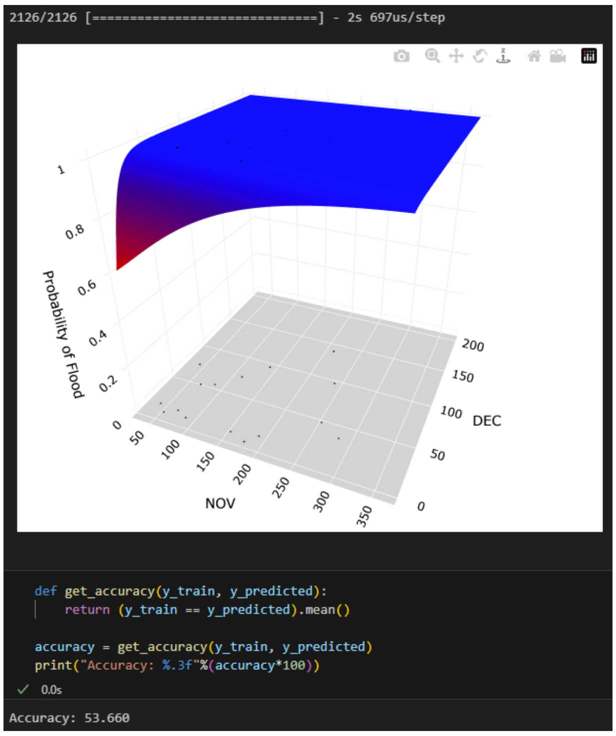
Table 1, Average accuracy of the FFNN model of 2-2-1 after running 10 times.

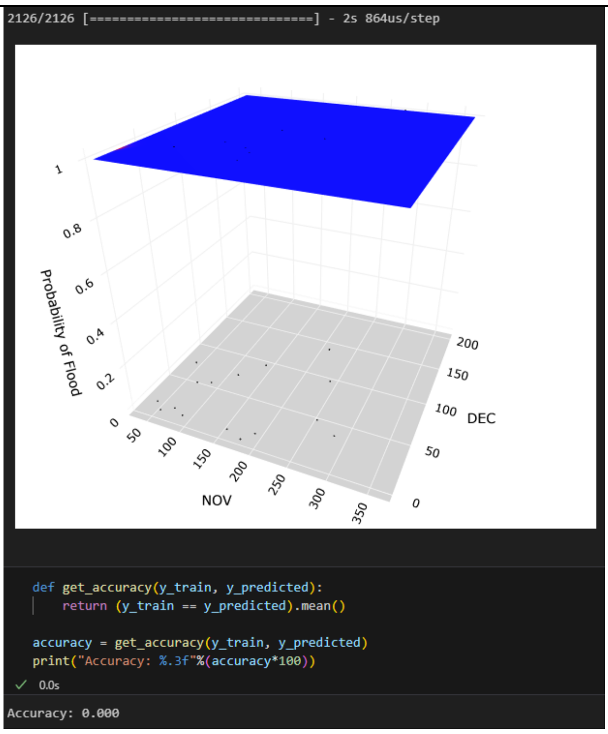
Screenshot output	Accuracy (%)

 `2126/2126 [==============================] - 2s 767us/step` `def get_accuracy(y_train, y_predicted):`
`    return (y_train == y_predicted).mean()`
 	
`accuracy = get_accuracy(y_train, y_predicted)`
`print("Accuracy: %.3f"%(accuracy*100))`
`✓ 0.0s`
`Accuracy: 55.319`
	55.319%

 `2126/2126 [==============================] - 2s 704us/step` `def get_accuracy(y_train, y_predicted):`
`    return (y_train == y_predicted).mean()`
 	
`accuracy = get_accuracy(y_train, y_predicted)`
`print("Accuracy: %.3f"%(accuracy*100))`
`✓ 0.0s`
`Accuracy: 21.894`
	21.894%


`2126/2126 [==============================] - 2s 707us/step`
`def get_accuracy(y_train, y_predicted):`
`    return (y_train == y_predicted).mean()`
 	
`accuracy = get_accuracy(y_train, y_predicted)`
`print("Accuracy: %.3f"%(accuracy*100))`
`✓ 0.0s`
`Accuracy: 27.255`
	27.255%


`2126/2126 [==============================] - 2s 697us/step`
`def get_accuracy(y_train, y_predicted):`
`    return (y_train == y_predicted).mean()`
 	
`accuracy = get_accuracy(y_train, y_predicted)`
`print("Accuracy: %.3f"%(accuracy*100))`
`✓ 0.0s`
`Accuracy: 53.660`
	53.660%



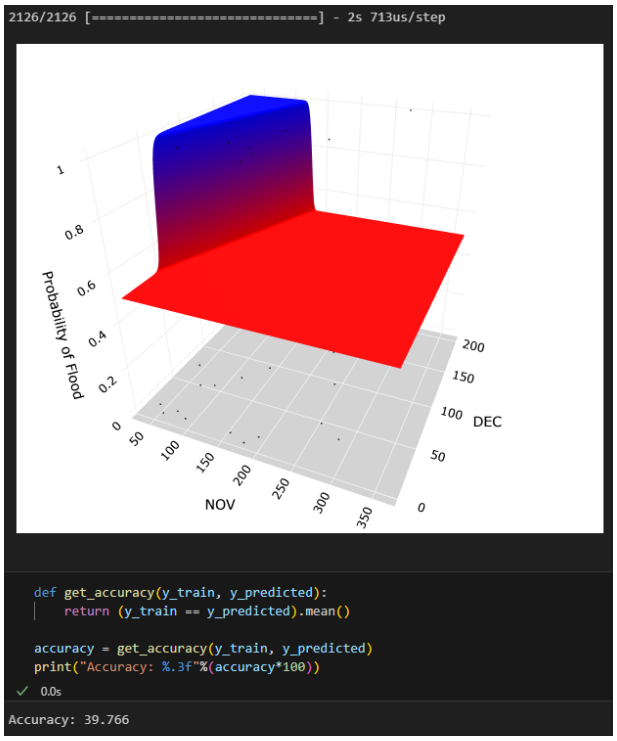
```
2126/2126 [==============================] - 2s 864us/step
```

```
def get_accuracy(y_train, y_predicted):
    return (y_train == y_predicted).mean()

accuracy = get_accuracy(y_train, y_predicted)
print("Accuracy: %.3f"%(accuracy*100))
```

✓ 0.0s

```
Accuracy: 0.000
```

0.000%



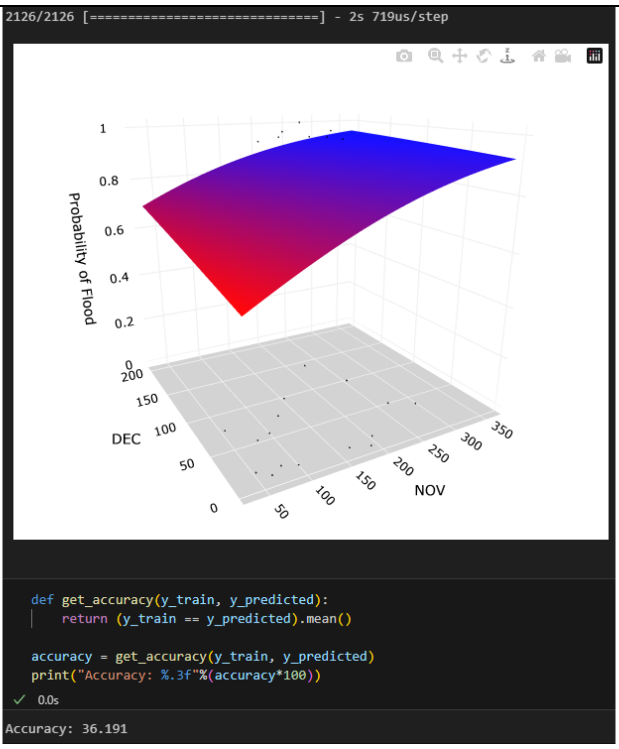
```
2126/2126 [==============================] - 2s 713us/step
```

```
def get_accuracy(y_train, y_predicted):
    return (y_train == y_predicted).mean()

accuracy = get_accuracy(y_train, y_predicted)
print("Accuracy: %.3f"%(accuracy*100))
```

✓ 0.0s

```
Accuracy: 39.766
```

39.766%



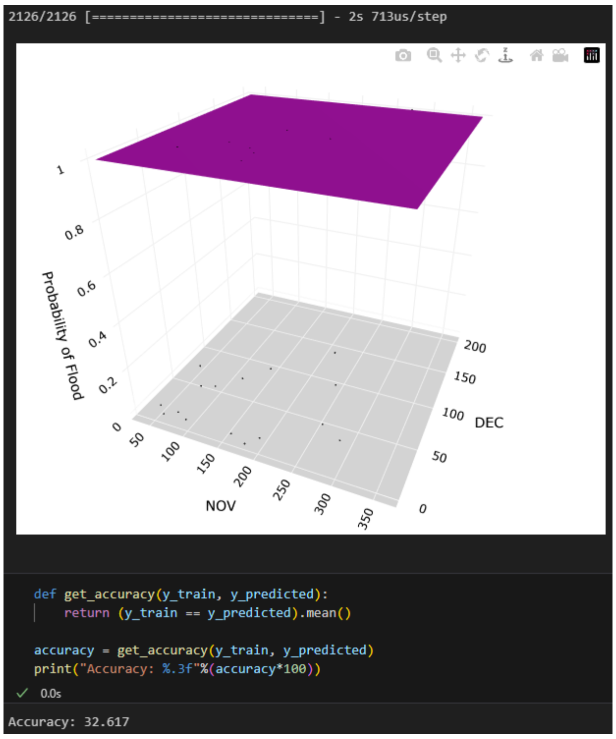
```
2126/2126 [==============================] - 2s 719us/step
```

```
def get_accuracy(y_train, y_predicted):
    return (y_train == y_predicted).mean()

accuracy = get_accuracy(y_train, y_predicted)
print("Accuracy: %.3f"%(accuracy*100))
```

✓ 0.0s

```
Accuracy: 36.191
```

36.191%



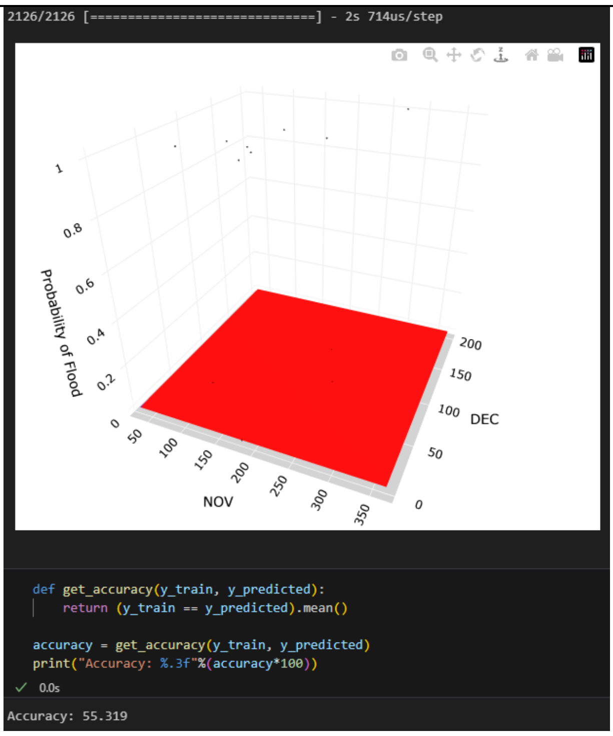
```
2126/2126 [==============================] - 2s 713us/step
```

```
def get_accuracy(y_train, y_predicted):
    return (y_train == y_predicted).mean()

accuracy = get_accuracy(y_train, y_predicted)
print("Accuracy: %.3f"%(accuracy*100))
```

✓ 0.0s

```
Accuracy: 32.617
```

32.617%



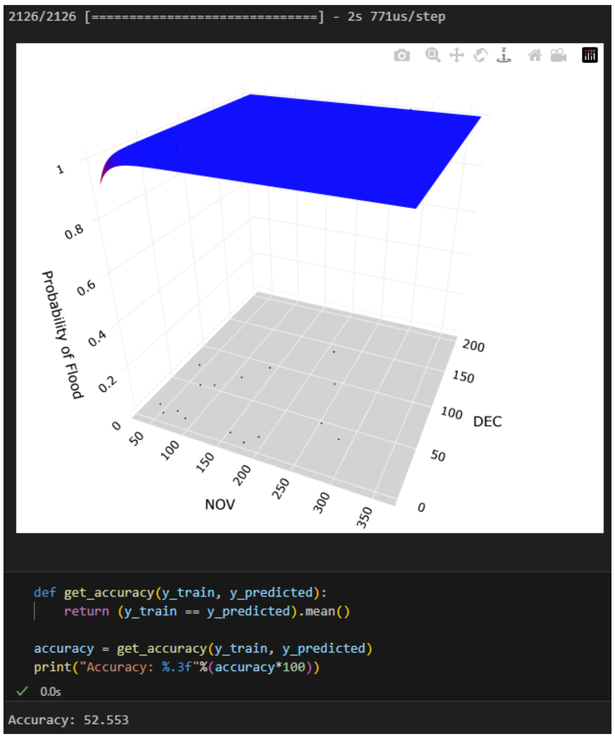
```
2126/2126 [==============================] - 2s 714us/step
```

```
def get_accuracy(y_train, y_predicted):
    return (y_train == y_predicted).mean()

accuracy = get_accuracy(y_train, y_predicted)
print("Accuracy: %.3f"%(accuracy*100))
```

✓ 0.0s

```
Accuracy: 55.319
```

55.319%



```
2126/2126 [==============================] - 2s 771us/step
```

```
def get_accuracy(y_train, y_predicted):
    return (y_train == y_predicted).mean()

accuracy = get_accuracy(y_train, y_predicted)
print("Accuracy: %.3f"%(accuracy*100))
```

✓ 0.0s

```
Accuracy: 52.553
```

52.553%

It is observed that the following results produced an output accuracy of the following:

55.319% , 21.894% , 27.255% , 53.660% , 0.000% , 39.766% , 36.191% , 32.617% , 55.319% , 52.553%

Based on the experiments conducted, it is observed that the model can reach an accuracy highest, 55.319%. During the time of experiment, this was observed to be the highest thus far the accuracy has ever reached when running this model.

6.3 Experiments carried out for the PSO-ANN model.

For this experimental attempt, the Kerala dataset is used as an input to the PSO-ANN model to evaluate the accuracy this model can reach. The architecture of the neural network in this model is 4-2-2 which indicates that it is **one Input layer** with four nodes, **one Hidden layer** with two nodes, and **one Output layer** with two nodes respectively. The goal of the experiment is to see the accuracy based on the number of nodes in the hidden layer and if this factor changes the accuracy of the model itself. Therefore, a series of test whereby the number of nodes in the hidden layer has 1, 5, 10, 15, and 20 nodes in the hidden layer is carried out. The model was running 5 times in total with different number of nodes in the hidden layer. The number of input features in this experiment is 4.

Table 2, Accuracy of the PSO-ANN model after running 5 times with different numbers of nodes in the **HIDDEN** layer.

Number of nodes in the *hidden layer*.	**Accuracy (%)**
1	100.000
5	100.000
10	100.000
15	100.000
20	100.000

It is observed that the number of nodes in the hidden layer does not affect the accuracy results. For a fair experimental observation, all values of input, hidden and output layers will be altered for the purpose of a fair experimental result. Therefore, the next experimental step for the PSO-ANN model is to change the values in the output layer. A series of test whereby the number of nodes in the output nodes has 3, 6, 9, 12, and 15 nodes is carried out. The model was also then running a total of 5 times with the respective number of nodes in the output layer. The number of input features in this experiment is 4.

Table 3, Accuracy of the PSO-ANN model after running 5 times with different numbers of nodes in the **OUTPUT** layer.

Number of nodes in the *output layer*.	**Accuracy (%)**
2	100.000
3	100.000
6	100.000
9	100.000
12	100.000
15	100.000

It is observed that the number of nodes in the output layers also does not affect the accuracy results. However, the node value cannot be equal to 1 as it will be out of bound since the model has an axis of 1 with size 1. The model needs 2 output nodes since the target data of the output has two outputs which is YES and NO which was then renamed to 1 and 0 respectively.

With the obtained results, the experimental phase can proceed to the next step, which is to alter the input values of the input layer. However, there are extra steps in this experimental procedure for the input layer. Since the value of the input layer's node will be altered, the input data of the dataset also must be altered since this is the layer where the data from the dataset enters to evaluate the entire model's accuracy results. Therefore, based off the input layer nodes that will be changed, the dataset input shapes will also be altered accordingly. For example, if the node of the Input layer is 5, the shape of the input data will also have to be reshaped to 5. In the next experiment procedure, several number of nodes will be the inputs in the input layer. The values of 1 to 8 will be used in this experiment as input features to observe the difference of accuracy. Then the model will perform extra runs but this time with a node value of 11 and 13 (13 input features which signifies the number of months (12) in a year plus the annual(1) rainfall).

Table 3, Accuracy of the PSO-ANN model running with different number of nodes in the **INPUT** layer.

Number of nodes in the *input layer*.	**Accuracy (%)**	**MSE**	**RMSE**
1	100.000	0.000	0.000
2	100.000	0.000	0.000
3	100.000	0.000	0.000
4	100.000	0.000	0.000
5	80.000	0.200	0.447
6	83.333	0.167	0.408
7	85.714	0.143	0.378
8	87.500	0.125	0.354
11	90.909	0.091	0.302
13	84.615	0.154	0.392

7.0 Results and Discussions

7.1 Accuracy performance of the PSO-ANN model with the FFNN model.

Table 4 shows the results obtained when the PSO is used to train the ANN using the Kerala flood dataset. The FFNN model where the neural network is trained using backpropagation using the Kerala dataset as well.

Table 4, Accuracy comparisons of Particle Swarm Optimization (PSO) in Training ANN and Feedforward Neural Network (FFNN)

Number of nodes in the input layer	Model	Accuracy (%)
1	PSO-ANN	100.000
2	PSO-ANN	100.000
3	PSO-ANN	100.000
4	PSO-ANN	100.000
1	FFNN	55.319
2	FFNN	55.319
3	FFNN	55.319
4	FFNN	55.319

With the usage of the Kerala flood dataset, the following architecture is used for the neural networks in the two models: for the PSO-ANN model, one input layer with two nodes, one hidden layer with two nodes, and one output layer with two nodes. For the FFNN model, one input layer with one node, one hidden layer with two nodes, and one output layer with one node. The FFNN model has two sets of architecture of the neural network. The second one consists of one input layer with two nodes, one hidden layer with four nodes, and one output layer with one node. Outside of the experimenting of the PSO-ANN model, the number of nodes in the input layer greater than 4 was not used since the PSO-ANN model provides very high accuracy with 1 to 4 number of nodes. The original plan was to use up to 10 nodes in the input layer, but even by doing so, the PSO-ANN would still produce better accuracy than the FFNN model.

However, the tables are turned in terms of accuracy when there are eleven nodes in the input layer. It is observed that PSO-ANN is performing worse compared to the highest possible accuracy outcome when compared to the FFNN model. But the drawback is that the FFNN is highly inconsistent and therefore needs to run several times like the experiments recorded in Table 1 to get the highest possible accuracy that the model could achieve whilst the PSO-ANN model only needs to run once. These results are obtained via the usage of the training data to predict for the best possible results and highest possible accuracy.

In order to have a fair comparison, three input features will be used for both the PSO-ANN model and the FFNN model for comparisons of performance in terms of accuracy. The months of October, November, and December has been selected as the input features from the Kerala dataset to be used in the training of both models to produce an accuracy output. The results are compared and highlighted in Table 4 as the outcome of the experiment.

7.2 Accuracy performance of the PSO-ANN model with the Benchmark 5 machine learning algorithms model.

Table 5 shows a comparison in terms of accuracy results obtained from the PSO-ANN model and the benchmark 5 machine learning algorithms model where the algorithms used are the K Nearest Neighbour algorithm (KNN), Support Vectors Classification algorithm (SVC), Logistic Regression algorithm (LOG), Decision Tree Classifier algorithm (DTC), and Random Forest Classifier algorithm (RFC).

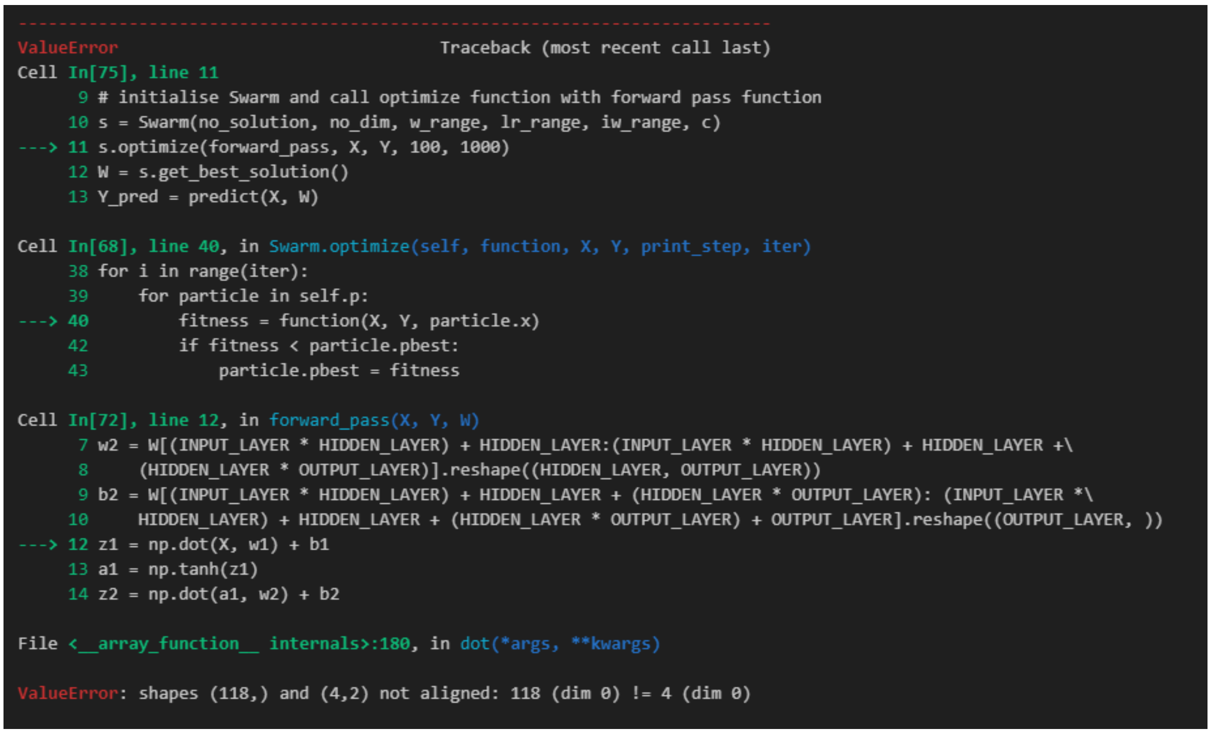
Table 4, Accuracy comparisons of PSO-ANN model and the Benchmark 5 machine learning algorithms model

Models	**Accuracy (%)**	**MSE**	**RMSE**
PSO-ANN	100.000	0.000	0.000
KNN	83.333	0.167	0.408
SVC	83.333	0.167	0.408
LOG	33.333	0.667	0.816
DTC	79.167	0.208	0.456
RFC	66.667	0.333	0.577

All accuracy tabulated are results using training data when running the model to evaluate the performance. The results compared in terms of accuracy whereby it is observed that the highest performance accuracy is the PSO-ANN. All accuracy results are based on a single dataset input into the models which is the Kerala flood dataset csv file. Such a comparison, however, may not be the best of results and therefore serves as a comparison of benchmark between the efficiency of the PSO training ANN model and the supervised machine learning algorithms. A better approach would be to have more swarm intelligence algorithm in training the neural network to produce the accuracy result. The accuracy results would then be compiled and tabulated to be compared with one another for a better study for which algorithm has the best optimization in terms of prediction with the supervised learning method.

8.0 Failures of experiments

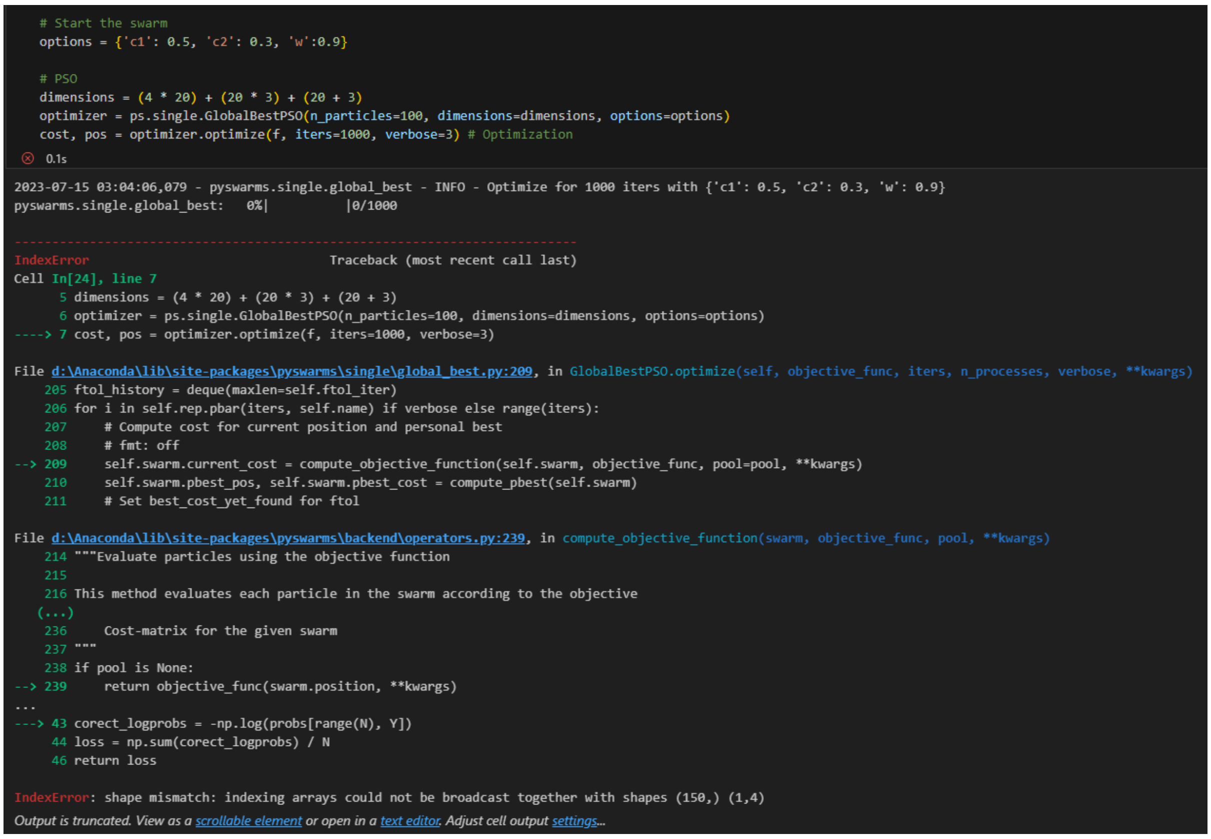
In this section, a discussion of the failures of attempts that has been carried out in this experiment thus far. This discussion will also highlight the possible outcomes if the failures were to be solved. One of the failures is the scalability of the dataset into the PSO model. This is due to the shape incompatibility of the data when initializing the optimization for either the test data or the training data and therefore does not align.



```
---------------------------------------------------------------------------
ValueError                                Traceback (most recent call last)
Cell In[75], line 11
      9 # initialise Swarm and call optimize function with forward pass function
     10 s = Swarm(no_solution, no_dim, w_range, lr_range, iw_range, c)
---> 11 s.optimize(forward_pass, X, Y, 100, 1000)
     12 W = s.get_best_solution()
     13 Y_pred = predict(X, W)

Cell In[68], line 40, in Swarm.optimize(self, function, X, Y, print_step, iter)
     38 for i in range(iter):
     39     for particle in self.p:
---> 40         fitness = function(X, Y, particle.x)
     42         if fitness < particle.pbest:
     43             particle.pbest = fitness

Cell In[72], line 12, in forward_pass(X, Y, W)
      7 w2 = W[(INPUT_LAYER * HIDDEN_LAYER) + HIDDEN_LAYER:(INPUT_LAYER * HIDDEN_LAYER) + HIDDEN_LAYER +\
      8     (HIDDEN_LAYER * OUTPUT_LAYER)].reshape((HIDDEN_LAYER, OUTPUT_LAYER))
      9 b2 = W[(INPUT_LAYER * HIDDEN_LAYER) + HIDDEN_LAYER + (HIDDEN_LAYER * OUTPUT_LAYER): (INPUT_LAYER *\
     10     HIDDEN_LAYER) + HIDDEN_LAYER + (HIDDEN_LAYER * OUTPUT_LAYER) + OUTPUT_LAYER].reshape((OUTPUT_LAYER, ))
---> 12 z1 = np.dot(X, w1) + b1
     13 a1 = np.tanh(z1)
     14 z2 = np.dot(a1, w2) + b2

File <__array_function__ internals>:180, in dot(*args, **kwargs)

ValueError: shapes (118,) and (4,2) not aligned: 118 (dim 0) != 4 (dim 0)
```

In this case of out experimental model, it produces a value error where it is indicating the shape is not aligned. The solution would be to normalise the input features to fit the data into the model for training and therefore must be reshaped for the alignment to proceed with the results.

Another failure during the process of making the model was when after an attempt of running the model, the output results were not achievable due to an error involving the shape output again. The error this time was that of an Index error whereby the shape has a mismatch when trying to produce an output.

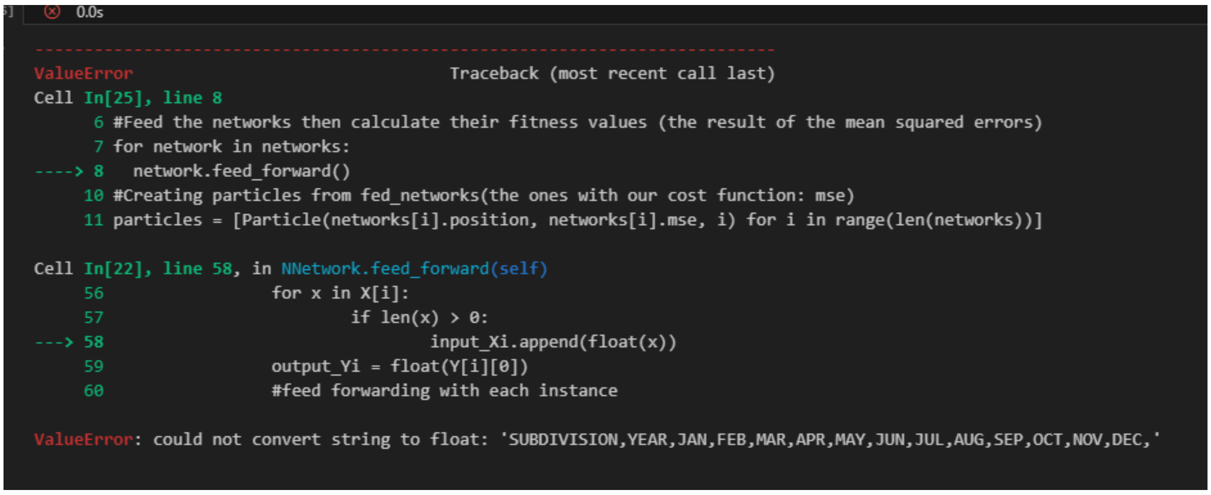


```
# Start the swarm
options = {'c1': 0.5, 'c2': 0.3, 'w':0.9}

# PSO
dimensions = (4 * 20) + (20 * 3) + (20 + 3)
optimizer = ps.single.GlobalBestPSO(n_particles=100, dimensions=dimensions, options=options)
cost, pos = optimizer.optimize(f, iters=1000, verbose=3) # Optimization
```

```
⊗ 0.1s

2023-07-15 03:04:06,079 - pyswarms.single.global_best - INFO - Optimize for 1000 iters with {'c1': 0.5, 'c2': 0.3, 'w': 0.9}
pyswarms.single.global_best:   0%|          |0/1000

---------------------------------------------------------------------------
IndexError                                Traceback (most recent call last)
Cell In[24], line 7
      5 dimensions = (4 * 20) + (20 * 3) + (20 + 3)
      6 optimizer = ps.single.GlobalBestPSO(n_particles=100, dimensions=dimensions, options=options)
----> 7 cost, pos = optimizer.optimize(f, iters=1000, verbose=3)

File d:\Anaconda\lib\site-packages\pyswarms\single\global_best.py:209, in GlobalBestPSO.optimize(self, objective_func, iters, n_processes, verbose, **kwargs)
    205 ftol_history = deque(maxlen=self.ftol_iter)
    206 for i in self.rep.pbar(iters, self.name) if verbose else range(iters):
    207     # Compute cost for current position and personal best
    208     # fmt: off
--> 209     self.swarm.current_cost = compute_objective_function(self.swarm, objective_func, pool=pool, **kwargs)
    210     self.swarm.pbest_pos, self.swarm.pbest_cost = compute_pbest(self.swarm)
    211     # Set best_cost_yet_found for ftol

File d:\Anaconda\lib\site-packages\pyswarms\backend\operators.py:239, in compute_objective_function(swarm, objective_func, pool, **kwargs)
    214 """Evaluate particles using the objective function
    215 
    216 This method evaluates each particle in the swarm according to the objective
   (...)
    236     Cost-matrix for the given swarm
    237 """
    238 if pool is None:
--> 239     return objective_func(swarm.position, **kwargs)
...
---> 43 corect_logprobs = -np.log(probs[range(N), Y])
     44 loss = np.sum(corect_logprobs) / N
     46 return loss

IndexError: shape mismatch: indexing arrays could not be broadcast together with shapes (150,) (1,4)
Output is truncated. View as a scrollable element or open in a text editor. Adjust cell output settings...
```

Due to time constraint, this model was then abandoned and marked as a failure attempt as it took too long in finding a solution to the error.

The last failure to be addressed would be a failure whereby the model that was developed is a model that works with unlabelled data. Therefore, it was a model for unsupervised learning. Therefore, when the model was input with the supervised dataset, a value error occurs indicating the failure conversion of string to float when attempting a generation of output.



```
⊗ 0.0s

---------------------------------------------------------------------------
ValueError                                Traceback (most recent call last)
Cell In[25], line 8
      6 #Feed the networks then calculate their fitness values (the result of the mean squared errors)
      7 for network in networks:
----> 8   network.feed_forward()
     10 #Creating particles from fed_networks(the ones with our cost function: mse)
     11 particles = [Particle(networks[i].position, networks[i].mse, i) for i in range(len(networks))]

Cell In[22], line 58, in NNetwork.feed_forward(self)
     56                 for x in X[i]:
     57                         if len(x) > 0:
---> 58                                 input_Xi.append(float(x))
     59                 output_Yi = float(Y[i][0])
     60                 #feed forwarding with each instance

ValueError: could not convert string to float: 'SUBDIVISION,YEAR,JAN,FEB,MAR,APR,MAY,JUN,JUL,AUG,SEP,OCT,NOV,DEC,'
```

Therefore, this model was also a failure and was dropped once finding out the model's function to unsupervised dataset and requires one dimensional raw data that is unlabelled.

9.0 Conclusion and Future Work

Swarm Intelligence is an effective optimization entity in the world of machine learning to solve global min-max problems with its application to real life scenarios. In this study, the methods where it involves the usage of swarm intelligence and the ones that do not involve swarm intelligence was examined and the results were compared with one another. The methods that do not involve swarm intelligence serves as a benchmark model to be compared with the swarm intelligence model. The proposed method has produced high accuracy when compared to several of the benchmark models that were used in this project. It has been observed that a swarm intelligence algorithm in training artificial neural network is a very effective method in works that involves predictive results.

Flash flood is well known for its destructive nature and its capabilities of performing large scale environmental damage. This natural disaster is also responsible for mankind's most devastating destructions that is recorded in the world history of natural disasters. In the view of climate changes, scenarios such as these becomes worse as the climatic events takes place in an unprecedented way. Prediction models are good sources or a tool in supporting flood disasters preparedness plan to eliminate or reduce the damage dealt to flood-prone areas. This project proposed an effective swarm intelligence algorithm in training artificial neural network for the highest possible accuracy in predicting flood. The experiments were done with a dataset of an urban area that was prone to flood and the results were promising.

This study in general has a lot of similar approaches and related works, however most of them involved are based on unlabelled datasets and therefore uses the unsupervised learning approach. The principles on the other hand, are quite similar in terms of accuracy measurement and benchmarking of several models in line with the proposed model. However, the true and real game changer in terms of data collections and experimental procedures are what decides the overall performance of a proposed model. Therefore, more effort and time should be spent with a more diversity of datasets and learning methods would be ideal. This method is a useful contribution in the evaluation of accuracy between models and its consistency with the output results. This current trend has a noticeable growth in the artificial intelligence industry where new optimization algorithms are appearing with promising results and is proving to be very reliable.

Regarding the future work of this study, in the future where more time permits, a remastered version of this study is something of a possible plan should the circumstances allow it. A lot of improvements to the methodology and applications can be made in this study for more data and results can be made. For instants, alongside Accuracy results, the results of ROC, and AUC could have also been included in this study when evaluating the performance of the models. Besides that, more swarm intelligence algorithms in training the artificial neural network can enter the picture to be compared alongside the benchmark models for better evaluation purposes. Improvements such as mentioned above can make the models more reliable should it be considered a plausible option in real-world application and more complex or huge datasets.

10.0 References

Bradford, A. (2022). What Is Artificial Intelligence, and How Does It Affect Your Daily Life? *Reader's Digest*. Retrieved from https://www.rd.com/article/what-is-artificial-intelligence/

Book from John H. Holland https://books.google.com.my/books?hl=en&lr=&id=5EgGaBkwvWcC&oi=fnd&pg=PR7&ots=mJlo70Njyi&sig=sIywE1sEjHpq2sBj3fiMeJdcTx8&redir_esc=y#v=onepage&q&f=false

Bui, Q.T., Nguyen, Q.H., Nguyen, X.L., Pham, V.D., Nguyen, H.D., Pham, V.M. (2020). Verification of novel integrations of swarm intelligence algorithms into deep learning neural network for flood susceptibility mapping. *Journal of Hydrology vol 581*. Retrieved from https://www.sciencedirect.com/science/article/pii/S002216941931114X.

Bui, D.T., Ngo, P.T.T., Pham, T.D., Jaafari, A., Minh, N.Q., Hoa, P.V., Samui, P. (2019). A novel hybrid approach based on swarm intelligence optimized extreme learning machine for flash flood susceptibility mapping. *Catena vol 179. Pg 184-196.* Retrieved from https://www.sciencedirect.com/science/article/pii/S034181621930147X

Copeland, B.J. (2022). Artificial Intelligence. *Britannica.com.* Retrieved from https://www.britannica.com/technology/artificial-intelligence

Chang, L.C., Chang, F.J., Yang, S.N., Kao, I.F., Ku, Y.Y., Kuo, C.L., Amin, I.M.Z. (2019). Building an Intelligent Hydro informatics Integration Platform for Regional Flood Inundation Warning Systems. *MDPI.* Retrieved from https://www.mdpi.com/2073-4441/11/1/9/htm

Darabi, H., Haghighi, A.T., Rahmati, O., Shahrood, A.J., Rouzbeh, S., Pradhan, B., Bui, D.T. (2021). A hybridized model based on neural network and swarm intelligence-grey wolf algorithm for spatial prediction of urban flood-inundation. *Journal of Hydrology vol 603*. Retrieved from https://www.sciencedirect.com/science/article/pii/S0022169421009045

Dongare, A.D., Kharde, R.R., Kachare, A.D. (2012). Introduction to Artificial Neural Network. *International Journal of Engineering and Innovative Technology (IJEIT).* Retrieved from https://citeseerx.ist.psu.edu/document?repid=rep1&type=pdf&doi=04d0b6952a4f0c7203577afc9476c2fcab2cba06

Fadlalah, S.O., Anderson, T.N., Nates, R.J. (2021). Artificial Neural Network-Particle Swarm Optimization (ANN-PSO) Approach for Behaviour Prediction and Structural Optimization of Lightweight Sandwich Composite Heliostat. *Springer Link*. Retrieved from https://link.springer.com/article/10.1007/s13369-021-06126-0

Geeksforgeeks. (2021). Introduction to Swarm intelligence. *Geeksforgeeks.org*. Retrieved from https://www.geeksforgeeks.org/introduction-to-swarm-intelligence/

JavaTpoint. (n.d). Artificial Neural Network Tutorial. *Javatpoint.com*. Retrieved from https://www.javatpoint.com/artificial-neural-network

JavaTpoint. (n.d). K-Nearest Neighbour (KNN) Algorithm for Machine Learning. *Javatpoint.com*. Retrieved from https://www.javatpoint.com/k-nearest-neighbor-algorithm-for-machine-learning

JavaTpoint. (n.d). Random Forest Algorithm. *Javatpoint.com*. Retrieved from https://www.javatpoint.com/machine-learning-random-forest-algorithm

Kadiyala, S.P., Woo, W.L. (2022). Flood Prediction and Analysis on the Relevance of Features using Explainable Artificial Intelligence. *ResearchGate.net*. Retrieved from https://www.researchgate.net/publication/357824114_Flood_Prediction_and_Analysis_on_the_Relevance_of_Features_using_Explainable_Artificial_Intelligence

Mosavi, A., Ozturk, P., Chau, K. (2018). Flood Prediction Using Machine Learning Models: Literature Review. *MDPI.* Retrieved from https://www.mdpi.com/2073-4441/10/11/1536/htm

Tam, A. (2021). A Gentle Introduction to Particle Swarm Optimization. *Machine Learning Mastery*. Retrieved from https://machinelearningmastery.com/a-gentle-introduction-to-particle-swarm-optimization/

State Level Bankers Comittee (N.D). Geographical Map of Kerala. *slbckerala.com.* Retrieved from https://slbckerala.com/geographical-map.aspx#:~:text=It%20lies%20between%20North%20latitudes,between%2011%20and%20121%20kilometres.

Schroer, A. (2022). Artificial Intelligence. *Builtin.* Retrieved from https://builtin.com/artificial-intelligence

Ngo, P.T.T., Hoang, N.D., Pradhan, B., Nguyen, Q.K., Tran, X.T., Nguyen, Q.M., Nguyen, V.N., Samui, P., Bui, D.T. (2018). A Novel Hybrid Swarm Optimized Multilayer Neural Network for Spatial Prediction of Flash Floods in Tropical Areas Using Sentinel-1 SAR Imagery and Geospatial Data. *MDPI.* Retrieved from https://www.mdpi.com/1424-8220/18/11/3704/htm

Panahi, M., Dodangeh, E., Rezaie, F., Khosravi, K., Le, H.V., Lee, M.J., Lee, S., Pham, B.T. (2021). Flood spatial prediction modelling using a hybrid of meta-optimization and support vector regression modelling. *Catena vol 199.* Retrieved from https://www.sciencedirect.com/science/article/pii/S0341816220306640

ScienceDirect. (2020). Swarm Intelligence. *Swarm Intelligence for Resource Management in Internet of Things.* Retrieved from https://www.sciencedirect.com/topics/engineering/swarm-intelligence#:~:text=Swarm%20intelligence%20is%20defined%20as,other%20based%20on%20simple%20principles.

Tayfur, F., Singh, V.P., Moramarco, T., Barbetta, S. (2018). Flood Hydrograph Prediction Using Machine Learning Methods. *MDPI.* Retrieved from https://www.mdpi.com/2073-4441/10/8/968/htm

Kurama, V. (2022). Feedforward Neural Networks: A Quick Primer for Deep Learning. *Builtin.com.* Retrieved from https://builtin.com/data-science/feedforward-neural-network-intro

World Health Organization (WHO). (n.d). Floods. Retrieved from https://www.who.int/health-topics/floods#tab=tab_1

Zehra, N. (n.d). Prediction Analysis of Floods Using Machine Learning Algorithms (NARX & SVM). *International Journal of Sciences: Basic and Applied Research (IJSBAR).* Retrieved from https://core.ac.uk/download/pdf/287366682.pdf