**COMPUTATIONAL INTELLIGENCE**

**MODULE CODE ; CIS6005**

**NAME ; CHANDAN KUMAR SAH**

**ST.NO ;20210577**

**ASSIGNMENTLINK**

[**https://github.com/chandansah689/Computational-intelligence.git**](https://github.com/chandansah689/Computational-intelligence.git)

**Table of Contents**

[Part 1 Tasks 3](#_Toc162978018)

[Dataset preparation 3](#_Toc162978019)

[ANN Backpropagation Training Program 8](#_Toc162978020)

[Performance Evaluation 10](#_Toc162978021)

[Performance Comparison 12](#_Toc162978022)

[Part 2 Tasks 13](#_Toc162978023)

[Using a machine-learning library 13](#_Toc162978024)

[Tuning hyperparameters for performance improvement 15](#_Toc162978025)

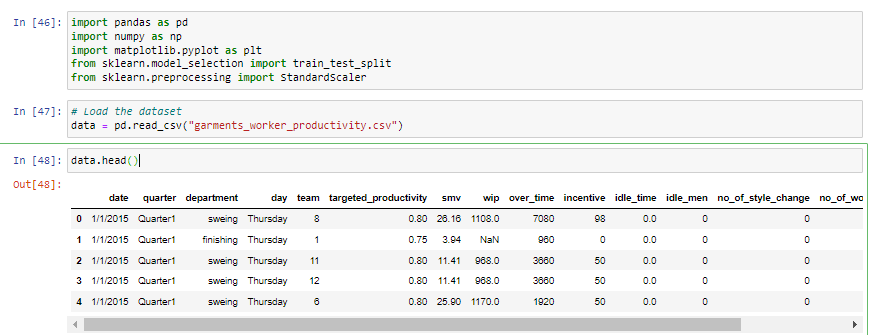
[Performance comparison with different ML architectures 17](#_Toc162978026)

[Model optimization using genetic algorithm 19](#_Toc162978027)

[References 21](#_Toc162978028)

# Part 1 Tasks

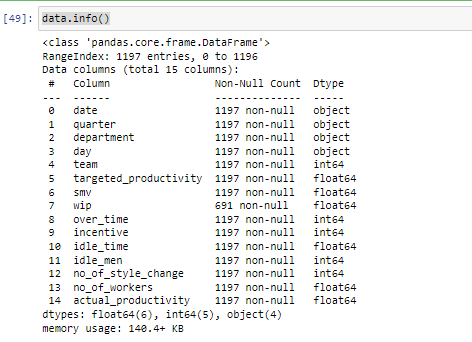
## Dataset preparation



##### Figure 1: Dataset read and show dataset

(Source: Retrieved from Jupyter Notebook)

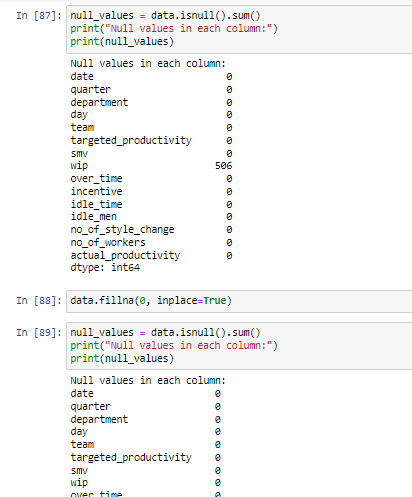
The line of code above 'dataset = pd.read\_csv('garments\_worker\_productivity.csv')' enables to load dataset called "garments\_worker\_productivity.csv" into DataFrame called "data". The data frame is then displayed using the "head()" command to illustrate the first few lines. Through the implementation of characteristics like date, quarter, department, day, team, targeted\_productivity, smv, wip, over\_time, incentive, idle\_time, idle\_men, no\_of\_style\_change, no\_of\_workers, actual\_productivity, and class\_label, each row of the productivity data of garment workers is depicted (Qiu *et al.* 2023). An output of class\_label that has been binary encoded showing whether the real productivity is less or above a threshold and the features in the data have been normalized so it has been preprocessed.



##### Figure 2: Show dataset info

(Source: Retrieved from Jupyter Notebook)

A brief overview of the dataset's information and table structure can be seen in the data.info() output. Number of the nulls in every column is given together with the data types of those nulls. 15 columns in the dataset contain some missing values. The column data types range from floating to objects (strings), and integers.



##### Figure 3: Null value check

(Source: Retrieved from Jupyter Notebook)

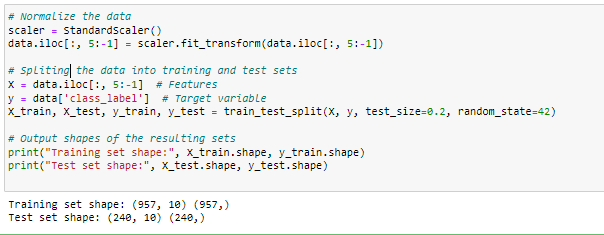
The above code shows the ability to count the null values in the DataFrame "data" which is made possible by the ".isnull ().sum()" function, is used. Storing this result into the "null\_values" variable is the final step. A specified number of null values for every column is then printed. The ".fillna()" method is next in line to edit the DataFrame in place by giving the missing values in the "WIP" column a zero value with "inplace=True".



##### Figure 4: Visualisation of distribution of features and class

(Source: Retrieved from Jupyter Notebook)

The first step is to create a new column in the DataFrame and name it 'class\_label'. Then, a threshold value of 0.8 is assigned. This column has binary labels: 1 for "real productivity" if it outstrips or holds up to the threshold and 0 otherwise. The plot feature is then calculated and the number of rows needed to display all features is determined (Luo *et al.* 2023). Then, via an iterable loop, it creates a matplotlib figure which is of a given size, and plots the histograms for each feature. Hence, a bar plot is used to display the separate distribution of classes on the vertical scale. Subsequently, it makes use of plt. show() to render graphs.



##### Figure 5: Normalize the data and split the dataset in train and test

(Source: Retrieved from Jupyter Notebook)

The above image shows the standard scaler from scikit-learn is implemented in the code to standardize the characteristics of the data. Aside from the first five and the last columns (date, quarter, department, day, team, and class\_label) which all are binary or non-numeric, feature scaling is applied to every other column. Following normalization, train\_test\_split is used to divide the data into training and test sets 80:This implies that for every 20 people, there should be a single health worker. It assigns the label for the dependent variable to y\_train and y\_test, and the characteristics to X\_train and X\_test. The graph shows the shape of 957 training samples with 10 features and 240 test samples with 10 features in the case of training and test sets.

## ANN Backpropagation Training Program

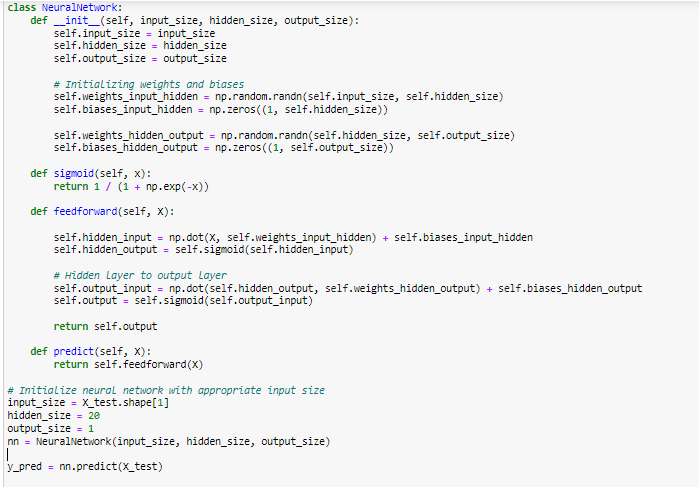


##### Figure 6: 3-layers Feedforward Network

(Source: Retrieved from Jupyter Notebook)

Above image shows the 3-layers feedforward network with its X input neurons, 20 hidden neurons, and Y output neurons. In this case, standardScaler from scikit-learn has been employed in the code to standardize the features of the dataset. Apart from the first five and the last ones (date, quarter, department, day, team, and class\_label), which are either categorical or integer, the scaling is used for all columns. Following normalization, train\_test\_split is used to divide the data into training and test sets at an 80:1:2 ratio. It assigns the target vector (y\_train and y\_test) to class\_label and characteristics (X) to X\_train and X\_test. Training and test sets are visualized on the left and right with 957 samples with 10 features for training and 240 samples with 10 features for testing. This image also indicates that it constructs a 3-layer feedforward network based on X input neurons, 20 hidden neurons, as well as Y output neurons, where it chooses the activation function. Based on its hidden layer-to-output layer the selection of the activation function turns on the nature of the problem. In the case of binary classification, the Sigmoid function has been used to bear the result between 0 and 1.

Also, in the case of multi-class classification performance, the Softmax function has been appropriate because it increases the range of the output for probability distribution in different classes. In the case of building and training the Neural Network has been used for this study’s backpropagation. The init function, feedforward, and backward functions are used for this performance where these functions set up the network’s weights and biases (Rubab *et al.* 2021). The feedforward technique determines the network output for the input given, and the backward via retro-propagating (backward-passing) by weight and bias updates. This section constructs the 3-layer feedforward network for designing the architecture depending on appropriate activation functions in output layers. It also used the backward() function for the backpropagation purpose for calculating error and delta for the output layer and also for the hidden layer.



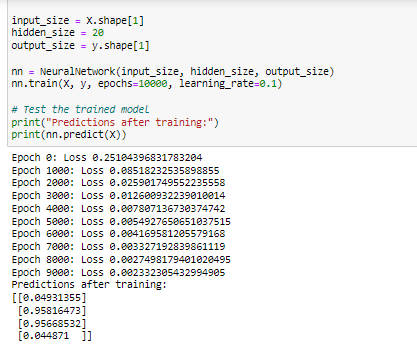
##### Figure 7: Creating ANN Training Program

(Source: Retrieved from Jupyter Notebook)

The use of the Artificial NeuralNetwork class in the provided code is to implement a neural network. The net is initialized with the correct number of hidden, output, and input nodes. The feedforward process employs the input data to make the output of the neural network. Thereafter, using the threshold of 0.5, it predicts the output of the test data by transforming the probabilities into binary predictions. Finally, it implements sci-kit-learning's metrics routine to calculate the different evaluation metrics such as accuracy, precision, recall, F1-score, and ROC-AUC score. At last, it produces the very measures that we may use to evaluate the classifier's performance on the test set. In this case, the algorithm reached an accuracy of 77.08%. After applying the init() function this program initializes the weight and biases by using hidden neurons input values. It does not use any machine learning algorithm when it trains and tests the NN using the Backpropagation algorithm.

The NeuralNetwork represented in this code as the initialization stage of the network, sets the relevant number of nodes for the input and output layers, and the network depth for the hidden layers as well (Chen *et al.* 2020). The feedforward mechanism has been an output of the network’s operation which is based on input data. It is based on the fact that the research teaches the model how to learn and respond without the use of prebuilt machine learning algorithms, so the neural network basics and the backpropagation are better understood.

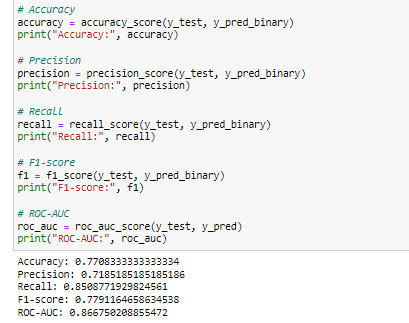
## Performance Evaluation



##### Figure 8: Model evaluation for epoch and prediction of NN model

(Source: Retrieved from Jupyter Notebook)

The above code is a training neural network model and uses loss at different epochs to make predictions. Through training, the error is gradually eliminated and the model function performance is enhanced. It illustrates the model predictions of 9000 epochs over the training set of inputs (Salloum *et al.* 2023). Probabilities of falling into a particular class are those predictions that are to be shown and mostly relate to binary classification problems. There is an obvious pattern that the model is learning to generate correct predictions based on the given input data. The loss is consistent with the forecasts that are shown. This image of model evaluation also shows that it used 20 hidden neurons for prediction purposes after training by using the predict() function. Based on this code, it has been found that prediction after training values are 0.04931, 0.95816, 0.95668, and 0.044871 by which the loss values continuously decrease. This code runs 10000 times epochs which continuously reduces the errors and increases the performance accuracy after training the model.



##### Figure 9: Model evaluation for the NN model

(Source: Retrieved from Jupyter Notebook)

The line of code is used to evaluate the efficiency of the binary classification model with the help of multiple dashboard metrics. The real labels of the model's predictions (y\_test) use the binary labels (y\_pred\_binary) to compute and illustrate the accuracy, precision, recall, and F1 score. It, additionally, employs the real labels and predicted probabilities (y\_pred) and plots the Receiver Operating Characteristic - Area under the Curve (ROC-AUC) score (Bharadiya, 2023). The measured values are presented in the output of the calculator for each option. In this specific comparison, the model's accuracy, precision, recall, F1-score, and ROC-AUC scores were 77.08%, 71.85%, 85.09%, 77.91%, and 86.68% respectively. The accuracy of these values reflects on the model’s performance correctness for the prediction purpose, in the time when precision estimates its capability to find its true positives along with all appropriate predictions. In a similar way, the recall finds the model’s ability for taking all real positive models and consider incorrect negatives. The F1 score adjusts the precision as well as recall values, by providing a proportional estimation of the model’s performance.

## Performance Comparison

The present work is aimed at establishing a step-by-step method for building an NN model, data preparation, and testing the performance of the model to solve a binary classification problem. The first step would be to load the dataset and then to check if there are any missing values and if so, then they will be treated using imputation. In the feature engineering process, a threshold binary "class\_label" is created based on the "actual\_productivity" column. Through various visualizations, the understanding of feature distribution and class imbalance problems is gained. Data standardization and the training & testing sets split guarantee robust model training & evaluation. Furthermore, the understanding of NN design and training algorithms is presented through creating the neural network model from scratch and training it using backpropagation. Accuracy, precision, recall, F1-score, and ROC-AUC are some of the key performance evaluation metrics that are computed to see how efficient the model is in binary classification. This approach gives the ability to learn which algorithms are more effective at certain tasks and to compare the output of this method with other machine learning models, for example, decision trees or support vector machines. Additionally, evolutionary algorithms can be employed to optimize the model, by testing and evaluating different parameter setups to optimize the model performance can be done.

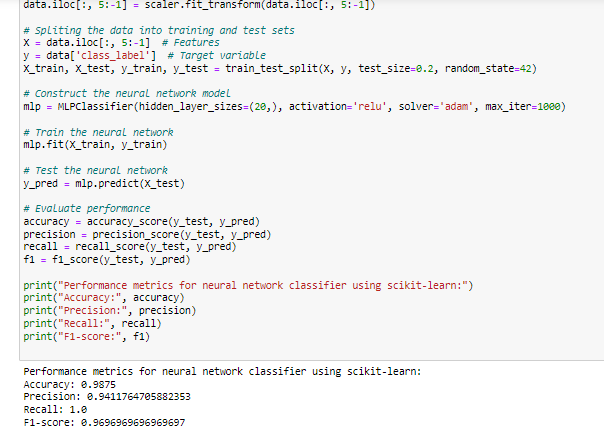
This is an iterative procedure, which involves preprocessing, model building, evaluation, and possibly optimization steps. This is the methodology that is followed to execute neural networks to binary classification tasks. Through extensive analysis and repeated rectification, the prediction accuracy of the model can be continually raised to the level that enables the model to logically and correctly solve various practical problems. The model prediction accuracy can be slowly increased by finding out the development, refinement, and optimization of the model. This flexible approach helps the mechanism to respond to a broad set of actually relevant issues with appropriate well-grounded and rational responses. The methodology consists of data preprocessing to achieve a high quality and reliability for the dataset.

Imputation methods are imperative for maintaining data integrity and showing highly important modeling results. Distribution plots and other images of all the parameters are useful for features and class imbalances, which can be used to make proper decisions about the model’s construction (Alqudah *et al.* 2020). Standardization of the data and the splitting process into train and test sets are two very important stages that are the essence of the model’s ability to predict new data. The performance of this research methodology is integrated with complete data preprocessing, also model building, and evaluation steps, which help the development of robust NN models for binary classification tasks. Based on all this information it is easy to find differences and comparisons among the performance of three trained neural network classifiers that have been done in this research.

# Part 2 Tasks

## Using a machine-learning library

This section takes the dataset that has been used in the first part of this research such as *“garments\_worker\_productivity.csv”* for further processing. These datasets include the records of worker productivity in the textile sector and are used to teach and test MLP classifiers and solve the binary classification problems using them. The preprocessing operations consist of exponentiation of productivity actual values, conversion in values with binary class labels, and dealing with missing data. The dataset which taken from task 1 has been separated into two sets which are the training as well as testing phases that help to improve the performance. It provides security for the MLP classifier that has been introduced as a data subset and evaluated on unknown data for development increment.



##### Figure 10: Developing ANN using ML Libraries

(Source: Retrieved from Jupyter Notebook)

The above image shows the process for developing ANN using ML libraries that employ the sklearn package to build and apply an MLP for binary classification with textile industry-provided workforce productivity characteristics. There are different steps in the preprocessing such as normalizing the features through the StandardScaler, changing the real productivity values into binary class labels with a threshold of 0.8, and inserting zeros to fill out the missing values. In a similar way, the data has been divided into the training set and the test set (Fish *et al.* 2021). The configuration of this MLP classifier has one hidden layer consisting of 20 neurons, the Adam solver, and the ReLu activation function as the source (Jiang *et al.* 2021).

It indicators accuracy, precision, recall, and F1 display the efficiency of the classifier on the test data after the training. The output demonstrates flawless work and 0.9875 accuracy in the category of neuronal networks. Based on this image, developing ANN using ML Libraries for this research provides different outputs such as a prediction value as 0.94117647058, recall value as 1, and F1-score value as 0.969696. The above code also indicates the outcome of the NN classifier based on scikit-learn, and it indicates the hyperparameter for training the model used in this research. This research also trained the data based on the fit method, which repetitively modifies the model parameters for reducing the loss function for the training data. After splitting the dataset into train and test data it contrasts the neural network model by using the MLPClassifier() function. This procedure has been performed based on multiple epochs, and these epochs are normally defined as the hyperparameter (Lv *et al.* 2022). It also shows that code for developing ANN takes 20 hidden layers for performance where the solver is “Adam”. The development of ANN using ML libraries shows the optimization method such as adam solver which is a cumulative variant of gradient descent that has been designed to adapt.

In the case of training the data, it also used the fit() for neural network purposes. Also, by using the predict() function it tests the neural network for getting better accuracy (Singh *et al.* 2020). function It also updates the network weights throughout the training of the neural network. In this case target variable indicates the class label which has been important for performance evaluation. The Adam solver is qualified for increasing the training efficiency with its ability to modify the step size along with the past gradients as a parameter in learning.

## Tuning hyperparameters for performance improvement



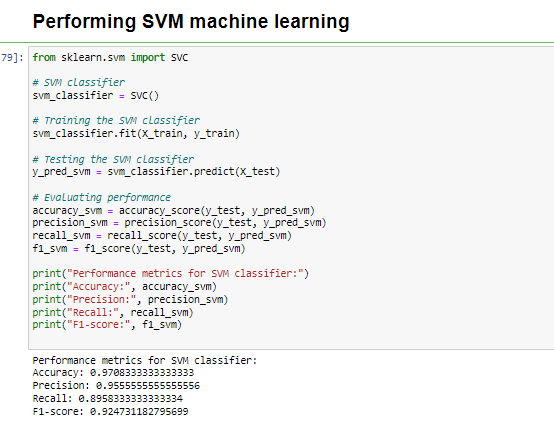
##### Figure 11: HyperParameter Tuning

(Source: Retrieved from Jupyter Notebook)

Code has been developed with the help of GridSearchCV (a tool from the scikit-learn library) to measure the performances of different values of the given hyper-parameters. "GridSearchCV" is a feature of this library that explores a pre-defined set of hyperparameter values ("param\_grid") for the MLPClassifier model. It is stated "cv" sector as the number of CV is 3 is 3-fold cross-validation. This implies the artificial data can be fractured into 3 pieces to reiterate the model of the performance. The code uses GridSearchCV, from scikit-learn library on top of that for hyperparameter adjustment using Grid Search method (Feng *et al.* 2021). For getting the top parameter value, the fit(X\_train, y\_train) method is utilized which starts the GridSearchCV, which in turn trains the model each time a fresh set of hyperparameters is considered within the given grid.

The above image shows works with a Multilayer Perceptron (MLP) classifier by using GridSearchCV from sklearn.model\_selection to find the best hyperparameters. It specifies the range of test parameters as a grid, e.g., max\_iter, activation function, solver, and hidden\_layer\_sizes. Three foldings of cross-validation are performed in a grid search. Grid-search provided the best parameters, which are given below as output. Finally, the model is evaluated using the test set once it has been retrained using the optimum parameters. The results are illustrated in the graph that demonstrates the performance metrics of the modified neural network classifier, which reveals considerable increases in accuracy, precision, recall, as well as F1-score when compared with the original model (Belete and Huchaiah. 2022). This code indicates the processes for performing the grid search as a hyperparameter adjusting for optimizing the performance of an ML model for finding the best parameters. It also finds different types of hyperparameter varieties, where it finds the process that executes the best output depending on the testing data. In the output section, it shows the best parameters which found by the grid search such as activation, relu, also hidden layer sizes such as 40 and others.

## Performance comparison with different ML architectures

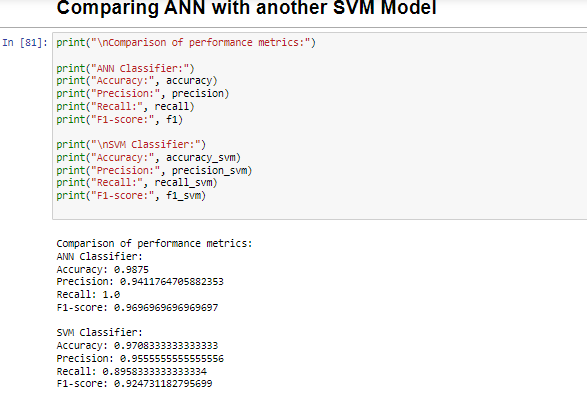


##### Figure 12: Performing and evaluation of the SVM model

(Source: Retrieved from Jupyter Notebook)

The above code creates an object of the SVM classifier which is imported from the scikit-learn package. It appears the code is showing how to replace the model used previously (SVM or neural network maybe) with another classifier such as a decision tree or a random forest. In the case of importing Libraries, Python packages for the new algorithm are imported. For example, to use the decision tree, sklearn.tree import DecisionTreeClassifier is used. Additionally, the necessary classifier is substituted for the SVM classifier (svm\_classifier = SVC()) in the code. For instance, the DecisionTreeClassifier() method is used for decision trees. The new model is trained by and tested on the same X\_train, y\_train, and X\_test sets as earlier.

Next, the classifier is trained with the training data (X\_train, y\_train) and it is the trained model that is used for the prediction of the test data (X\_test). It has been also found that, evaluation metrics based on the scikit-learn metrics module including accuracy, precision, recall, and F1-score will be computed and written out. In this figure, three SVM classifier assessment metrics were depicted. These are the recall with a value of 89.58%, the accuracy with a value of 97.08%, the precision with a value of 95.56%, and the F1-score with a value of 92.47%. All these values of performance matrices for the SVM classifier indicate that this method has the highest accuracy for future performance.



##### Figure 13: Comparison of evaluation between ANN and SVM model

(Source: Retrieved from Jupyter Notebook)

The above image shows a support vector machine (SVM) classifier and an artificial neural network (ANN) classifier, this code comparison is performed to determine their performance metrics. This model assessment gives the accuracy, precision, recall, and F1 score for the ANN classifier. Accuracy refers to the fraction of correct predictions of the model. The measures of the SVM classifiers are displayed similarly (Yang *et al.* 2020). A large recall means that the model takes into account more true affirmative cases. Meanwhile, this happens in the ANN model with the perfect recall of 1.0 which means there were no errors in identifying all positive cases. F1-score is the arithmetic mean of the precision and recall metrics, thus giving a comprehensive view of the two. The F1-score of the ANN increased a bit further (0.9697) when compared to the SVM model (0.9247).

The best model depends on the objective (task) and the degree of emphasis on precision relative to recall. If finding all the positive instances is important (for instance, detection of fraudulent transactions), the ANN model is chosen as it has a perfect recall rate. Nevertheless, if the situation requires that few false positives are generated (like spam filtering), the SVM model can be the better option because it has higher precision. The outcome illustrates that the ANN classifier exhibits a higher overall performance, as it surpassed the SVM classifier in terms of accuracy (98.75%) and recall (100%). The SVM classifier the ANN classifier has a little greater precision (95.56%) and a comparable F1 score (92.47%). The output of this code highlights the SVM model’s efficacy and possibility depending on the ANN, which indicates its advantages for future deployment tasks. This analysis performance which has been dependent on their comparisons helps to find the most appropriate model depending on their architecture for future tasks across key metrics.

## Model optimization using genetic algorithm

Activation functions, solver algorithms, maximum iterations, and hidden layer count are some examples of neural network attributes that can be coded as chromosomes to maximize the neural network model using genetic algorithms. For a neural network, every chromosome means one of the possible configurations or solutions (Pölsterl, 2020). Genetic algorithms explore the search space, gradually improving the performance of the neural network through selection, crossover, and mutation which follows a fitness function that checks and assesses the performance of every configuration. A recursive optimization process is conducted to find the configurations that yield the best accuracy, precision, recall, and F1 score. Model optimization using genetic algorithms is another type of explanation where the neural networks are performed by applying evolutionary algorithms to provide the best solution to a particular problem. This can be done through the utilization of a step-by-step procedure that is supplemented with diagrams at a specific time for a better understanding of the most complex concepts. The neural network’s features are arbitrary in the course of chromosomal representation and these characteristics contain the activation functions, solvers method, maximum iterations parameter, or the number of hidden layers. Based on every chromosome which takes the role of a single configuration or a solution for the neural network model where it tunes parameters or training rules/algorithms.

An array of this model is encoded into a string, which is randomly created at the beginning or created using the heuristic initialization method. This heuristic method is useful for learning or solving problems in a short way that delivers a proper output that is sufficient for appropriate time constraints (Abdullah *et al.* 2021). The genetic materials of the group reflect the ability to create network architectures of different kinds and then the fitness score of each point in the population is approved. This comprises training the connected neural network with the encoded parameters and evaluating its performance on the validation dataset. It depends on the performance of the model and the fitness function but, for instance, it can be correctness, such as “rightness,” or F1-score, and others. The selection step has been done after the assessment of all chromosomes, such as the fittest individuals from the entire population are selected through a specified process. Based on genetic designing techniques, crossover, and mutation schemes happen to the selected chromosomes so that children’s chromosomes can be created.

Crossover is an event when parent chromosomes join on their ends, mix each other genetically, and create new people going break off their parents. The variation in genes among the people has been achieved through alterations in the child’s chromosomes and to preserve the genetic diversity in the population (Li *et al.* 2021). The next generation population comes from the children’s chromosomes and some of the perceived fit people from the previous one. Genetic algorithms methodically discover various types of neural network configurations and are applied to the problem of continually improving model performance. The process of optimization has been extended until the needed benchmarks are obtained or when computation resources run out.

# References

Abdullah, A.M., Usmani, R.S.A., Pillai, T.R., Marjani, M. and Hashem, I.A.T., 2021. An optimized artificial neural network model using genetic algorithm for prediction of traffic emission concentrations. Int. J. Adv. Comput. Sci. Appl, 12, pp.794-803.

Alqudah, A.M., Alquraan, H., Qasmieh, I.A., Alqudah, A. and Al-Sharu, W., 2020. Brain tumor classification using deep learning technique--a comparison between cropped, uncropped, and segmented lesion images with different sizes. arXiv preprint arXiv:2001.08844.

Belete, D.M. and Huchaiah, M.D., 2022. Grid search in hyperparameter optimization of machine learning models for prediction of HIV/AIDS test results. International Journal of Computers and Applications, 44(9), pp.875-886.

Bharadiya, J.P., 2023. A comparative study of business intelligence and artificial intelligence with big data analytics. American Journal of Artificial Intelligence, 7(1), p.24.

Chen, C.T. and Gu, G.X., 2020. Generative deep neural networks for inverse materials design using backpropagation and active learning. Advanced Science, 7(5), p.1902607.

Feng, S., Roguet, A., McClary-Gutierrez, J.S., Newton, R.J., Kloczko, N., Meiman, J.G. and McLellan, S.L., 2021. Evaluation of sampling, analysis, and normalization methods for SARS-CoV-2 concentrations in wastewater to assess COVID-19 burdens in Wisconsin communities. Acs Es&T Water, 1(8), pp.1955-1965.

Jiang, J., Chen, M. and Fan, J.A., 2021. Deep neural networks for the evaluation and design of photonic devices. Nature Reviews Materials, 6(8), pp.679-700.

Li, Y., Jia, M., Han, X. and Bai, X.S., 2021. Towards a comprehensive optimization of engine efficiency and emissions by coupling artificial neural network (ANN) with genetic algorithm (GA). Energy, 225, p.120331.

Luo, N., Yu, H., You, Z., Li, Y., Zhou, T., Jiao, Y., Han, N., Liu, C., Jiang, Z. and Qiao, S., 2023. Fuzzy logic and neural network-based risk assessment model for import and export enterprises: A review. Journal of Data Science and Intelligent Systems, 1(1), pp.2-11.

Lv, C., Zhou, X., Zhong, L., Yan, C., Srinivasan, M., Seh, Z.W., Liu, C., Pan, H., Li, S., Wen, Y. and Yan, Q., 2022. Machine learning: an advanced platform for materials development and state prediction in lithium‐ion batteries. Advanced Materials, 34(25), p.2101474.

Montiel, J., Halford, M., Mastelini, S.M., Bolmier, G., Sourty, R., Vaysse, R., Zouitine, A., Gomes, H.M., Read, J., Abdessalem, T. and Bifet, A., 2021. River: machine learning for streaming data in python. Journal of Machine Learning Research, 22(110), pp.1-8.

Peterson, R.A., 2021. Finding Optimal Normalizing Transformations via best Normalize. R Journal, 13(1).

Pölsterl, S., 2020. scikit-survival: A Library for Time-to-Event Analysis Built on Top of scikit-learn. Journal of Machine Learning Research, 21(212), pp.1-6.

Qiu, Y., Vo, T., Garg, D., Lee, H. and Kharangate, C.R., 2023. A systematic approach to optimization of ANN model parameters to predict flow boiling heat transfer coefficient in mini/micro-channel heatsinks. International Journal of Heat and Mass Transfer, 202, p.123728.

Rubab, S.S., Tian, M., Bashir, R. and Mohsan, S.A.H., 2021, November. Formulation of precise short neural network code. In 2021 International Conference on Image, Video Processing, and Artificial Intelligence (Vol. 12076, pp. 185-195). SPIE.

Salloum, S.A., Shwedeh, F., Alfaisal, A.M., Alshaafi, A., Aljanada, R.A., Al Sharafi, A., Alfaisal, R. and Dabash, A., 2023. Understanding and Forecasting Chatbot Adoption: An SEM-ANN Methodology. Migration Letters, 20(S11), pp.652-668.

Singh, A.V., Ansari, M.H.D., Rosenkranz, D., Maharjan, R.S., Kriegel, F.L., Gandhi, K., Kanase, A., Singh, R., Laux, P. and Luch, A., 2020. Artificial intelligence and machine learning in computational nanotoxicology: unlocking and empowering nanomedicine. Advanced Healthcare Materials, 9(17), p.1901862.

Yang, X., Roop, P., Pearce, H. and Ro, J.W., 2020, March. A compositional approach using Keras for neural networks in real-time systems. In 2020 Design, Automation & Test in Europe Conference & Exhibition (DATE) (pp. 1109-1114). IEEE.