ISSN: 2754-6659

Journal of Artificial Intelligence & Cloud Computing

SCIENTIFIC Research and Community

Review Article

Open Access

Deep Learning Based Analysis of Student Aptitude for Programming at College Freshman Level

V Lakshmi Narasimhan[1] and G Basupi[2*]

[1]Georgia Southern University, USA

[2]University of Botswana, Botswana

ABSTRACT

Predicting Freshman student's aptitude for computing is critical for researchers to understand the underlying aptitude for programming. Dataset out of a questionnaire taken from various Senior students in a high school in the city of Kanchipuram, Tamil Nadu, India was used, where the questions related to their social and cultural backgrounds and their experience with computers. Several hypotheses were also generated. The datasets were analyzed using three machine learning algorithms namely, Backpropagation Neural Network (BPN) and Recurrent Neural Network (RNN) (and its variant, Gated Recurrent Network (GNN)) with K-Nearest Neighbor (KNN) used as the classifier. Various models were obtained to validate the underpinning set of hypotheses clusters. The results show that the BPN model achieved a high degree of accuracies on various metrics in predicting Freshman student's aptitude for computer programming.

***Corresponding author**

G Basupi, University of Botswana, Botswana.

Received: May 15, 2023; **Accepted:** May 23, 2023; **Published:** May 30, 2023

Keywords: Freshman Students, Aptitude for Programming, Machine Learning, K Nearest Neighbor (KNN), Backpropagation Neural Network (BPN), Recurrent Neural Network (RNN) and Gated Recurrent Unit (GNN).

Introduction

Computer science programs around the world always have students facing problems with programming courses, such as Programming Principles, Object-Oriented Programming and Data Structures. Most of the time, students fail later programming courses because of the lack of foundation at Freshmen level and, this has something to do with their aptitude for programing. In order to help potential applicants of the program to understand the kind of competencies of abstract and logical thinking skills that are needed for computer science programing courses and to help the University admissions office in selecting potential candidates for such courses, a study has been carried out among final year high school students. A model that helps in predicting student aptitude for programming based on several features was built. Datasets acquired through a questionnaire were analyzed using data analytics software, e.g., Python Libraries, PyTorch and TensorFlow. The model used machine learning to classify the student aptitude for computer programming, which helps in identifying students who are at risk of failing; note that improving the passing rates in introductory courses has a direct impact on retention rate also. Unlike other studies, which often correlate student's aptitude to programming to student's past academic performance, this study takes into account student's family background and individual's interaction with technology also; the authors feel that these are foundational factors in student's attitude to programming. The rest of the paper is organized as follows: section 2 describes the problem statement and related literature, while section 3 details the hypotheses and the questionnaire design. Section 4 presents the analysis approach & deep learning algorithms used, while section 5 provides the results of the study.

Problem Description & Related Literature

The literature review has two components, namely, i) those relating to model development and related factors that affect aptitude for programming and ii) the use of machine learning techniques for analyzing student's aptitude for programming. Identifying the factors that affect student aptitude to programming can also help us to understand how students learn to program, which in turn can help to plan the needed intervention at an early stage and avoid the risk of student retention in the program. Most of the factors that can be situated with one's aptitude for computer programming are abstract thinking, logical thinking, mathematical skills, and problem-solving skills. Some studies indicate that gender plays a role and in this modern age, males do dominant the world of computer Science. Results from several studies show that there is a statistically significant difference in programming performance between male and female students, where male students performed better than their female counterparts. A key factor that researchers omit is students background, but instead assume that student's performance is based on their previous grades or such skills as problem solving skills. Psychological and sociological factors play a big part in student's aptitude for programming and such factors as predictors can be helpful in understanding the process that students go through when learning (Longi, September 26, 2016). Other studies indicate that just the behavior of a student during lectures

and labs play a huge impact towards aptitude for programming – e.g., gestures, outbursts, and other factors including collaboration with other students (Ahadi & Lister, 2015). However, these studies lack depth of statistical significance in the larger context of the underlying question. A small amount of literature exists on using machine learning techniques for analyzing student's aptitude for programming. A Master's thesis (Longi, September 26, 2016) details the use of Bayesian network to model the relationship between factors that affect programming performance. Further detailed literature analysis can be obtained from the authors, which due to space limitations has been curtailed, but can be obtained from the authors.

Hypotheses & Questionnaire Design

In our earlier detailed research work [7], we developed a set of hierarchical hypotheses and corresponding questionnaire for comprehending Freshman aptitude for computer programming – as shown in Appendix–A. They deal with psychological and sociological aspects of student's views on computing and programming. It is noted that these hypotheses and questionnaire are better suited for Third World countries and rural students in the First World countries, who in general are not exposed to computing and computers much. The questionnaire was distributed to Senior students in a high school in the city of Kanchipuram, Tamil Nadu, India, and a large body of data was collected.

Dataset Description & Analysis Basics

It is noted that the questionnaire (vide. Appendix-A) is in two parts, viz., Part-A and Part-B. Details from the questionnaire were converted into a dataset format, which was further cleaned for any errors; missing values were normalized using mean values. The dataset consisted of 157 instances, with 35 attributes. The Likert Scale was used to rate each of the questions asked and it ranged from 0 to 7, where 0 is none, 1 – 6 being Strongly Disagree to Strongly Agree and 7 being not applicable. Part-A of the dataset deals with students' biographical backgrounds which are detailed in Fig.1 & 2.

Nature of Parameters, Attributes for Student-Biographical Information

	A	B	C	D	E	F	G	H	I	J	K	L	M
1	gender	age	Htown	GAI	Caste	AH-edu	P-situ	Medu	Fedu	Sedu	L-Access	AGmaths	AGphysic
2	1	1	0	2	[2,3]	5	3	[4,5]	[4,5]	[4,5]	5	2	2
3	1	0	Mawana	2	8	2	3	5	5	0	3	5	5
4	1	0	Meerut	2	8	2	3	6	0	0	6	5	4
5	1	1	Meerut	2	8	2	3	5	5	4	7	4	4
6	2	1	Mawana	0	8	2	3	5	5	2	4	5	5
7	1	1	Meerut	2	3	2	3	5	4	2	7	3	3
8	1	1	Mawana	2	3	2	3	5	5	4	1	5	4
9	1	1	Meerut	2	8	2	3	5	0	0	6	5	5
10	1	1	Mirarpur	0	8	2	3	4	4	2	7	4	4
11	2	1	Meerut	1	3	2	3	4	5	3	7	5	4
12	1	1	Meerut	1	3	2	1	3	4	2	7	2	2

Figure 1: Part-A Student-Biographical Information

(int) gender
(int)age
(text) Htown - Student's home town
(numeric) GAI - Student's gross annual income
(numeric) Caste - Student's Caste
(numeric) AH-edu - Student's Average Highest Education Level
(numeric) P-situ - parental living situation of student
(numeric) Medu - mother's education (numeric) Fedu - father's education
(numeric) Sedu - sibling's education
(numeric) L-Access - student's laptop access (numeric) AGmaths - last student's average
(numeric) AGphysics - last student's average physics exam
(text) NProg - students most used programming language
(bool) ProgVisual - Does student use visual programming tools
(bool) PairProg - Does student use pair programming tools
(numeric) AverageDistance - Students distance between home and institution
(numeric) Transport - Mode of transport of student
(numeric) ClassSize - Size of class
(numeric)MathsRepeat - # of times student repeated mathematics course

Figure 2: Dataset Attributes

Part-B of the dataset (Fig.3) relates ti students' programming experience, which is derived from clustering Q1-Q35 from Appendix-A. Each of the questions are evaluated using a Likert scale ranging from 0-7.

Attributes for Computer-Programming-Experience

	A	B	C	D	E	F	G	H	I	J
1	Q1	Q2	Q3	Q4	Q5	Q6	Q7	Q8	Q9	Q10
2	3	2	4	1	5	1	5	5	2	
3	0	0	0	0	0	0	0	0	0	
4	0	0	0	0	0	0	0	0	0	
5	1	2	3	4	1	2	3	4	1	
6	6	1	6	6	1	0	2	4	0	[2, 6]
7	1	2	3	4	5	6	7	6	5	
8	3	7	7	5	3	2	4	5	6	
9	0	0	0	0	0	0	0	0	0	
10	0	0	0	0	0	0	0	0	0	
11	6	1	6	1	1	7	6	1	6	
12	4	7	4	4	[2, 7]	[2,7]	2	5	5	
13	6	1	6	1	1	1	[6, 3]	[1, 6]	6	

Figure 3: Part B Computer-Programming-Experience Dataset based on multiple questions Q1 – Q35, each having a Likert range of 0 – 7.

Calculation of Mean, Median, Standard Deviation & Rho Values

This paper focusses on Part-B of the dataset which deals with calculating each of the features Mean, Median, Standard Deviation and RHO (Spearman's Correlation).

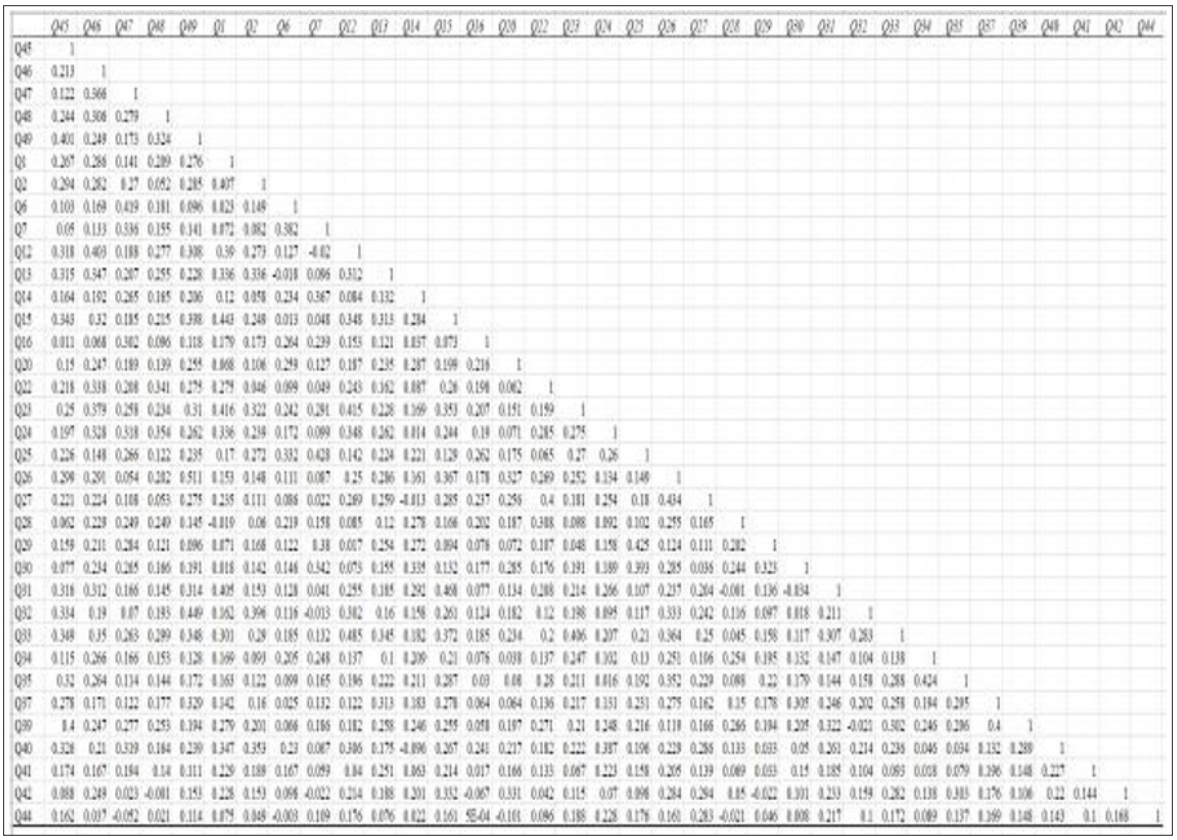

Figure 4: Correlation matrix of all the questions

Hypotheses Clustering

Appendix-A provides the list of hypotheses (H1-H10) and their grouping of the underlying questions into H0 to H10. Once grouped next was selection of the hypothesis were correlated to generate inferences. In this case C1 represents the correlation of hypotheses H0, H1, H2; C2 represents the correlation of H3, H6, H5; C3 represents the correlation of hypotheses H8, H8.1, H8.2 and, C4 represents the correlation of hypotheses H7, H9 and H10.

Analysis Approach & Deep Learning Algorithms Used

Fig.5 presents an overview of the workflow, whereas three different machine learning approaches were selected, namely, BNN and RNN (GRU) along with K-nearest neighbor algorithm for clustering; a brief overview of these techniques are presented in Appendix-B. Further, while data analytics open source libraries such as, PyTorch, WEKA, TensorFlow and RapidMiner are available, TensorFlow was chosen because: i) it provides excellent functionalities and services when compared to other popular deep learning , 2) has low-level libraries which provides more flexibility and 3) a highly interactive development environment that allows for design-as-you-go flexibility. Agile process model was used to develop Machine Learning based model/s as shown in Figure 4.

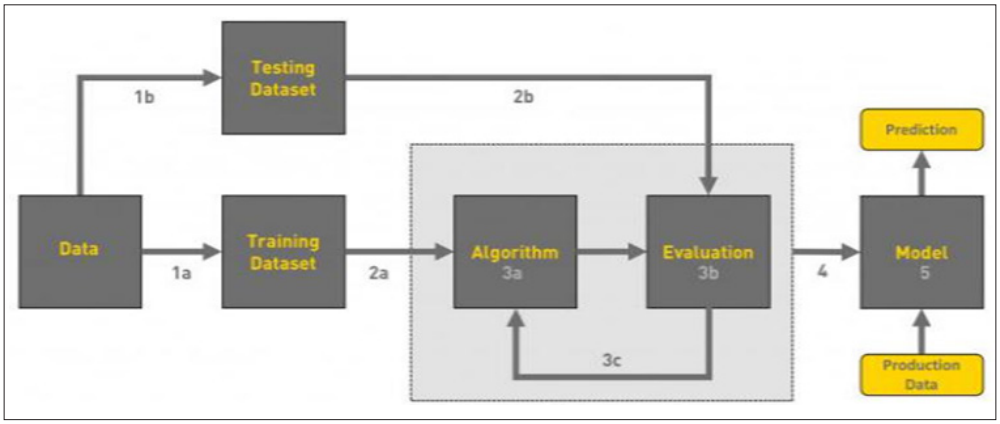

Figure 5: Overview of Workflow of Machine Learning

Analysis of Results

The models were evaluated using the performance metrics of accuracy, precision, recall, and F1- score, which are defined as follows:

- **Accuracy:** characterizes the degree to which a predicted value agrees with an actual value (Devasia & Vinushree, 2016).
- **Precision:** identifies the probability of a positive test result. High precision values indicate that the probability of the test set being accurately classified is high. In the context of this paper, precision indicates the number of students having aptitude for programming.
- **Recall:** evaluates the number of true positives of the actual class predicted by the models.
- **F1-score:** indicates the best performing algorithm.

Model Implemention Summary

Backpropagation Neural Network (BPN)

With the Backpropagation Neural Network (BPN), the datasets were randomly divided into training and test data in the ratio of 80:20; note that Ward, Peters, & Shelley (2010) state that if the size of the training dataset is too small or too large, the performance of the models will be affected. The output variables are the mean of each of the correlation regression. ADAM optimizer [10] was used for each model instead of the classical stochastic gradient descent procedure to update network weights. ADAM updates all parameters with individual learning rates so that every parameter in the network has a specific learning rate.

Correlation C1 relates to hypotheses H0, H1 and H2 and in this case the model was built with one input layer, two hidden layers and one output layer which took an input dimension of 7, 7 representing 7 features (or questions) based on the three hypotheses. The input layer contains 14 nodes with initial weight being uniform and activation function being Sigmoid [11]. The two hidden layers contain 8 nodes and 4 nodes respectively, both having initial weights being uniform and activation function being Sigmoid, while the output layer has one node.

The first test of the model using the number of epochs set to 100 yielded an accuracy of 37.5%, as shown in Fig.6. As the number of epochs increased from 100 to 400 and a change in the loss function from binary cross to MSE (Mean Square Error), the prediction accuracy became 94%; further the use of MSE was better than the binary cross function (Fig.7). However, the Loss graph still did not converge, thereby meaning that the model has not reached the stopping criteria. With 900 epochs, the model yielded a prediction accuracy of 93.75%. Based on the three tests, the loss graph starts converging at approximately 600 epochs. The stopping criteria of this model would possibly be at 600 epochs as anything beyond 900 gives an accuracy of 93.750% as can be evident from Figure 8; the accuracy remains constant from the 500 epochs and beyond. The corresponding Confusion matrix is shown in Figure 9.

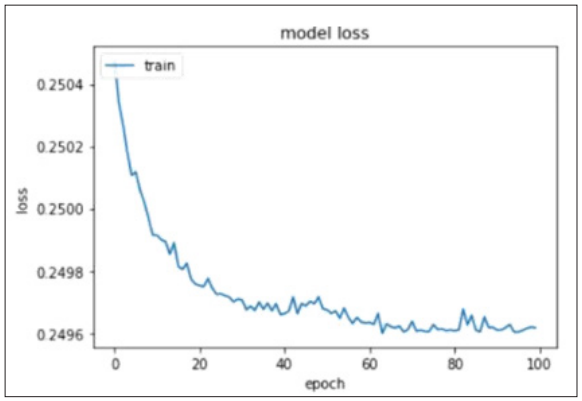

Figure 6: Model loss graph

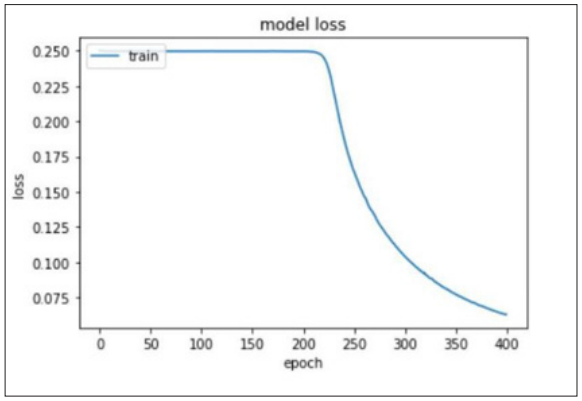

Figure 7: Model Loss Graph for 400 epochs

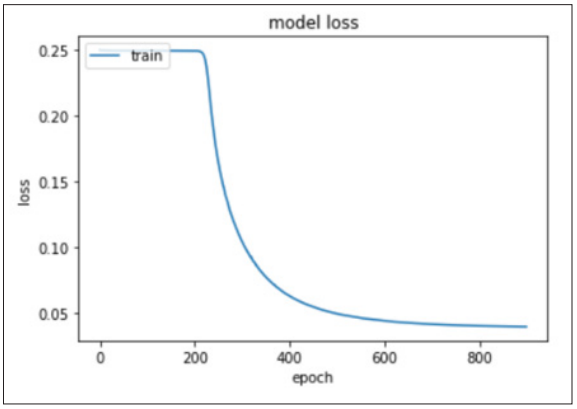

Figure 8: Model Loss Graph for 900 epochs

Confusion matrix

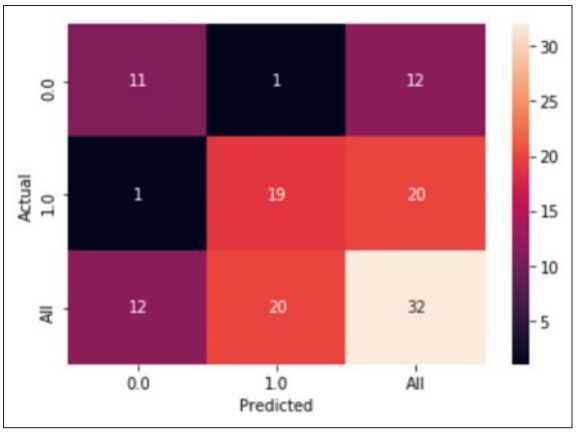

Figure 9: Confusion Matrix

TP = True Positives = 19
TN = True Negatives = 11
FP = False Positives = 1
FN=FalseNegatives=1

Correlation 2 relates to hypotheses H3, H5 and H6 and has seven input parameters and the co0orresponding model was built using one input layer, four hidden layers and one output layer. With 400 epochs along with Sigmoid activation function and MSE as the loss function, the model gave a prediction accuracy of 79.411765 %. With an increase in the number of epochs from 400 to 700 (Figs. 10 & 11), the model yielded an accuracy of 87.5%. With 1200 epochs, the Prediction accuracy is 90.625% and the corresponding confusion matrix as shown in Figure 12.

Correlation 3 relates to hypotheses H8.0, H8.1 and H8.2 and has seven input parameters. The model gave a prediction value of 96.875% with the test of 800 epochs (Figs. 13 & 14) and was declared to yield high prediction accuracy. The Confusion Matrix is shown in Fig.15.

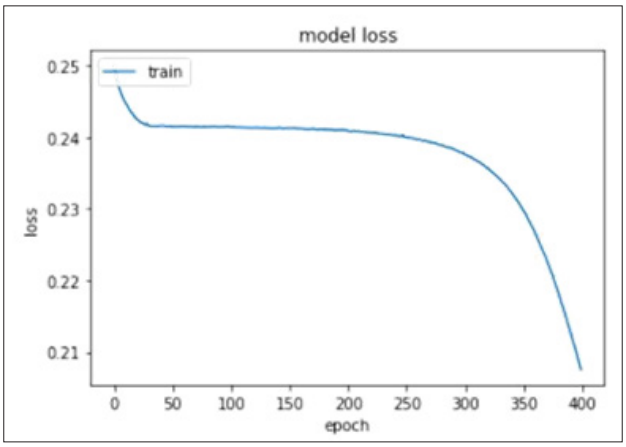

Figure 10: Model Loss Graph for 400 epochs

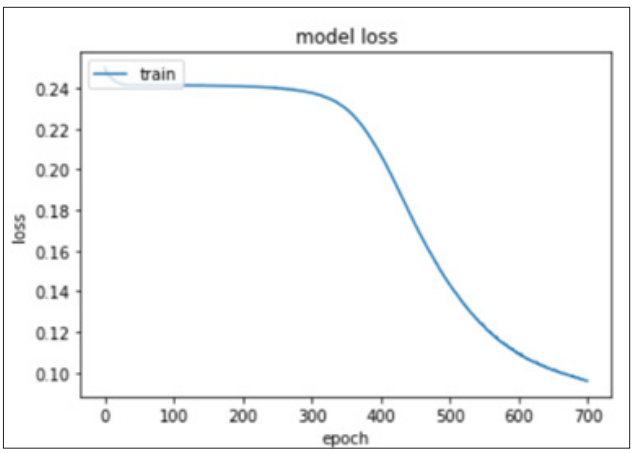

Figure 11: Model Loss Graph for 700 epochs

Confusion matrix

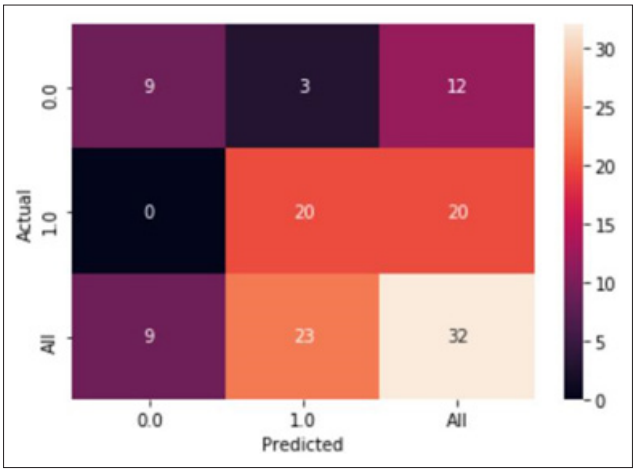

Figure 12: Confusion Matrix

TP = True Positives = 20
TN = True Negatives = 9
FP = False Positives = 3
FN = False Negatives = 0

Correlation 3 relates to hypotheses H8.0, H8.1 and H8.2 and has seven input parameters. The model gave a prediction value of 96.875% with the test of 800 epochs (Figs. 13 & 14) and was declared to yield high prediction accuracy. The Confusion Matrix is shown in Fig.15.

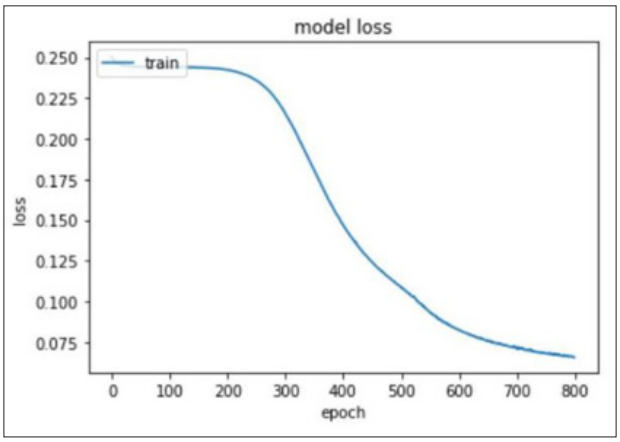

Figure 13: Model Loss Graph for 800 epochs

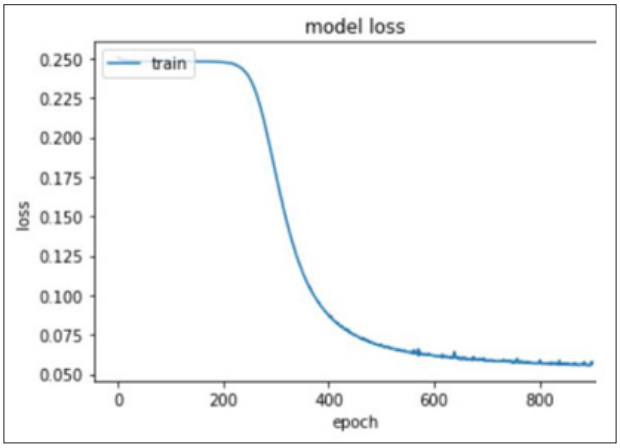

Figure 14: Model Loss Graph for 900 epochs

Confusion Matrix

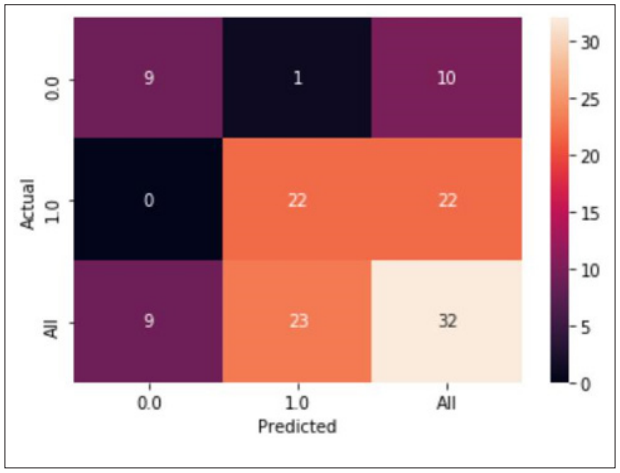

Figure 15: Confusion Matrix

TP = True Positives = 22
TN = True Negatives = 9
FP = False Positives = 1
FN = False Negatives = 0

Correlation 4 relates to hypotheses of H7, H9 and H10 and has nine input parameters and the related model is built using five layers using 900 epochs to train the model, yielding an accuracy of 90.625% (Figs. 16 & 17). The error loss reached the point of convergence with 900 epochs, because of the increase in the number of features. However, even with the addition of an extra layer, the accuracy remained the same hence no further layers are also required. The corresponding confusion matrix is shown in Figure 18.

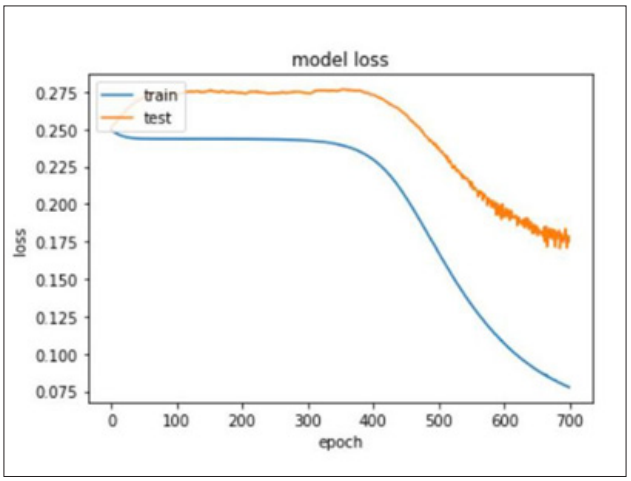

Figure 16: Model Loss Graph for 700 Epochs

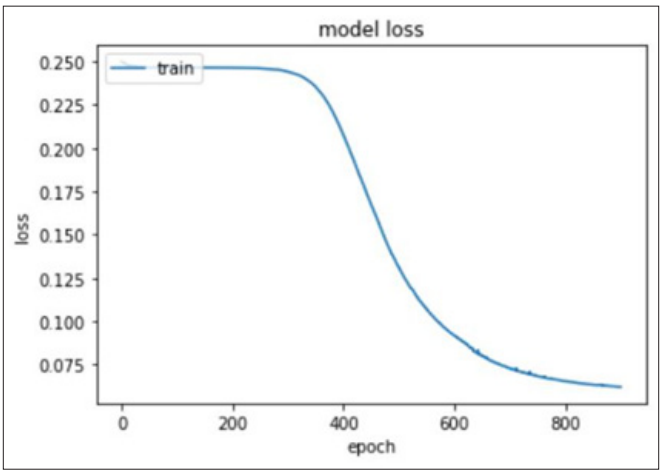

Figure 17: Model Loss Graph for 800 Epochs

Confusion Matrix

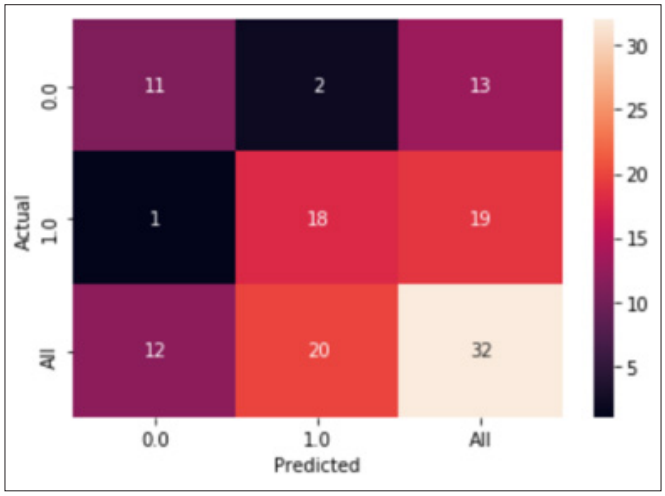

Figure 18: Confusion Matrix

TP = True Positives = 18
TN = True Negatives = 11
FP = False Positives = 2
FN = False Negatives = 1

Gated Recurrent Unit (RGU-ANN)
The datasets were randomly divided into training and test data on the ratio of 80:20. The output variables were the mean of each of the correlation regression and ADAM optimizer was used for each model. All the models were built with activation='Sigmoid' and loss='MSE' and with a batch size of 15.

Correlation 1 The GRU build model yielded an accuracy of 84.375% at 700 epochs.

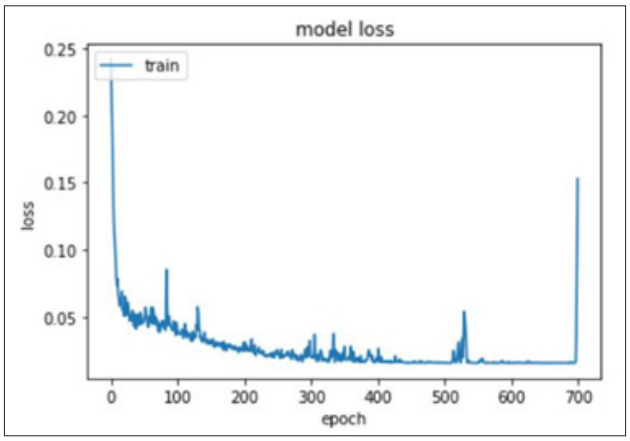

Figure 19: GRU Model Loss Graph for 700 epochs

It is evident from Figs. 19 & 20 that between 500 to 550 epochs, the model is overfitting and hence the number of epochs is now reduced, which in turn yielded an accuracy of 90.62500%. The corresponding Confusion Matrix is presented in Figure 21.

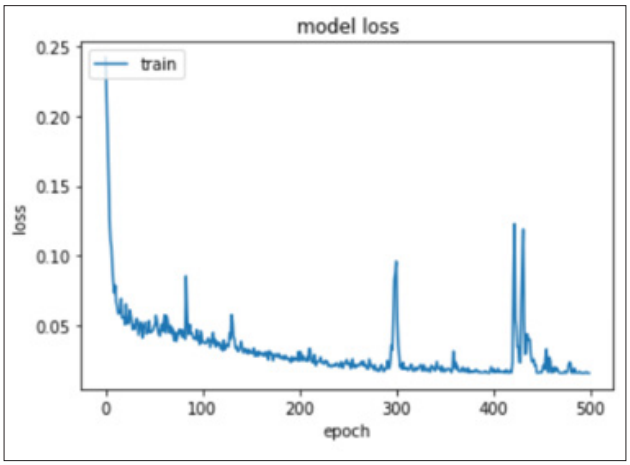

Figure 20: Model Loss graph for 500 epochs

Confusion Matrix

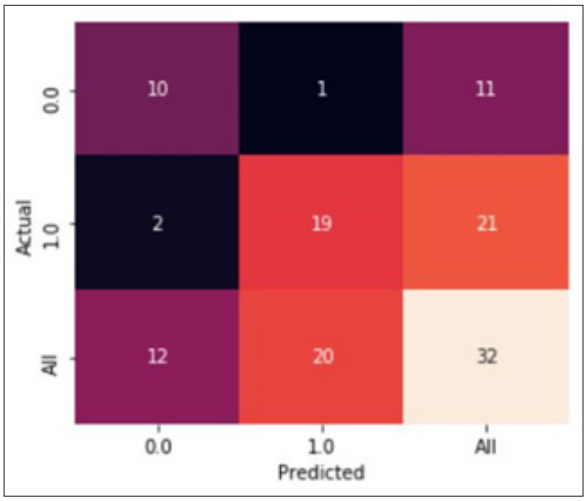

Figure 21: Confusion Matrix

TP = True Positives = 19
TN = True Negatives = 10
FP = False Positives = 1
FN = False Negatives = 2

Correlation 2 The model was built using 600 epochs as shown in Fig. 22 and the confusion matrix is shown in Fig. 23. The model gave a prediction accuracy of 84.37500%.

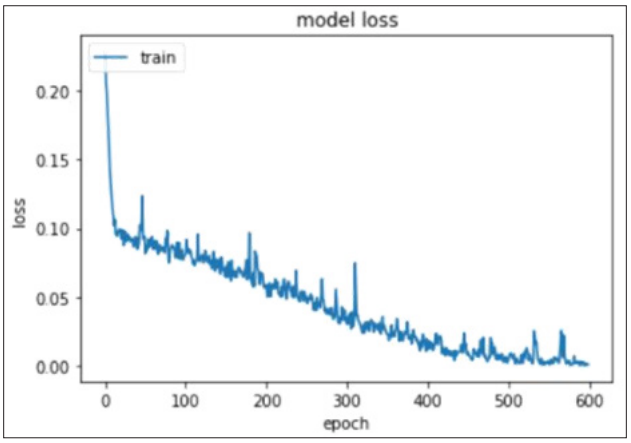

Figure 22: Model Loss for 600 epochs

Confusion Matrix

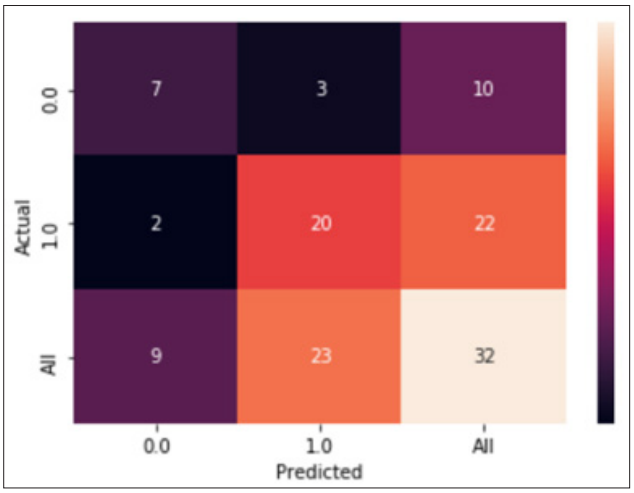

Figure 23: Confusion Matrix

TP = True Positives = 20
TN = True Negatives = 7
FP = False Positives = 3
FN = False Negatives = 2

On **Correlation 3**, the model yields a prediction accuracy of 96.87500% with the number of epochs set to 500;the model is declared as having the highest prediction accuracy (Fig.24); the corresponding confusion matrix is shown in Figure 25.

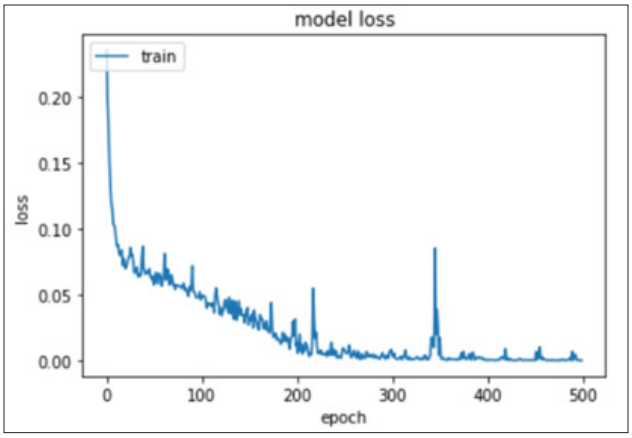

Figure 24: Model Loss for 500 epochs

Confusion Matrix

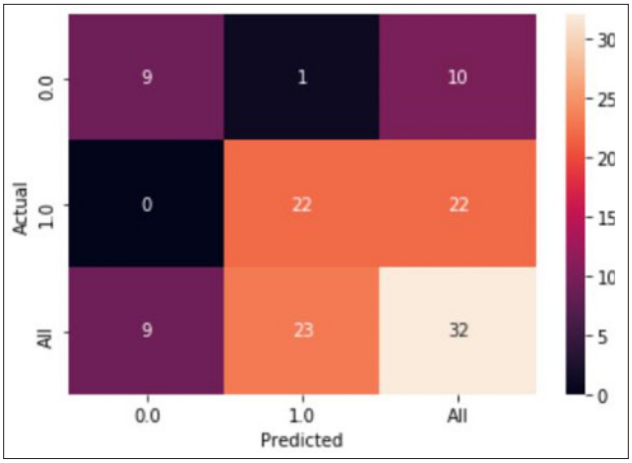

Figure 25: Confusion Matrix

TP = True Positives = 22
TN = True Negatives = 9
FP = False Positives = 1
FN = False Negatives

On **Correlation 4**, the model shown in Fig.26 yields a prediction value of 96.875% at 400 epochs, which was the highest accuracy; the confusion matrix is shown in Fig.27.

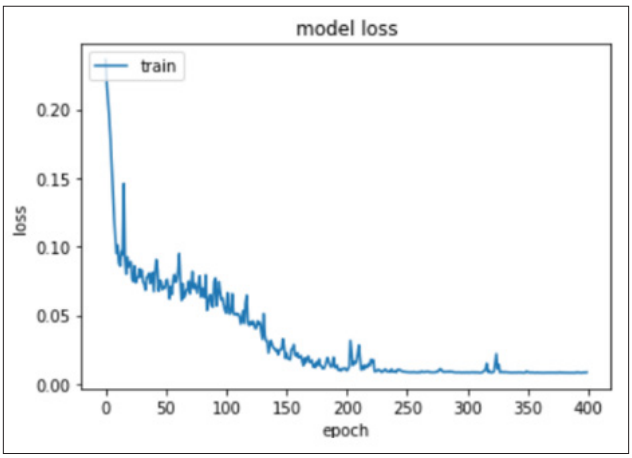

Figure 26: Model Loss for 400 Epochs

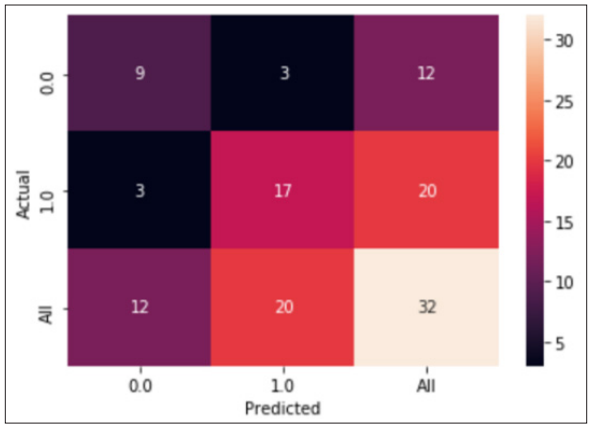

Figure 27: Confusion Matrix

TP = True Positives = 17
TN = True Negatives = 9
FP = False Positives = 3
FN = False Negatives = 3

K-Nearest Neighbor (KNN)
The machine learning algorithms were applied using hypothesis variables as input to predict aptitude for programming. The KNN was constructed using Python and Kera's library and the training and test data ratio was set at 80:20. The N-neighbor for each of the model was set to 5; note that the rule of thumb says K = Square Root of N divided by 2, where N is the number of samples in the training set.

For correlation 1, the model yields the best prediction accuracy of 84.375% at 5 nearest neighbors; the corresponding confusion matrix is shown in Fig.28.

Confusion Matrix

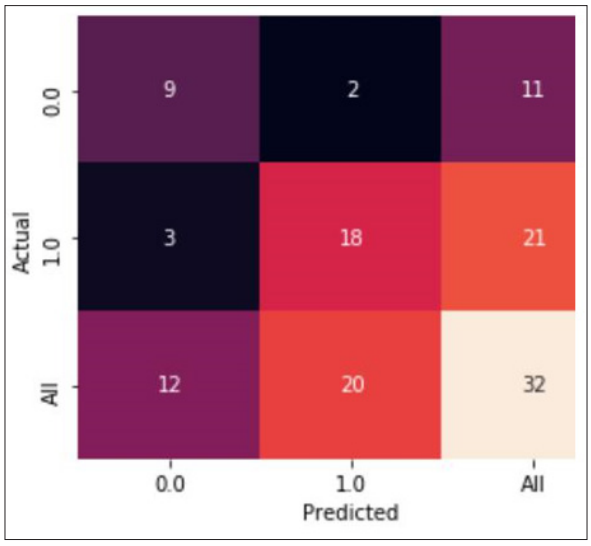

Figure 28: Confusion Matrix

TP = True Positives = 18
N = True Negatives = 9
FP = False Positives = 2
FN = False Negatives = 3

For correlation 2, the model yields a prediction accuracy of 75.%; the corresponding confusion matrix is shown in Fig.29.

Confusion Matrix

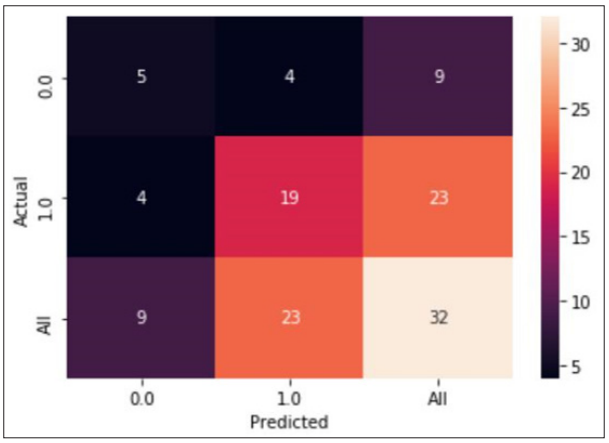

Figure 29: Confusion Matrix

TP = True Positives = 19
TN = True Negatives = 5
FP = False Positives = 4
FN = False Negatives = 4

For correlation 3, the model yields a prediction accuracy of 96.875%; the corresponding confusion matrix is shown in Fig.30.

Confusion Matrix

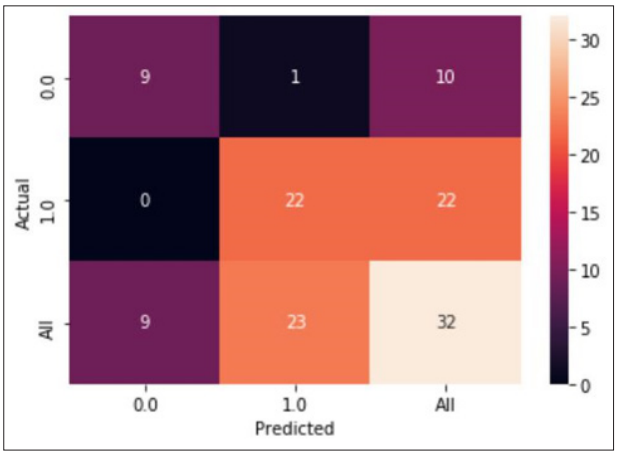

Figure 30: Confusion Matrix

TP = True Positives = 22
TN = True Negatives = 9
FP = False Positives = 1
FN = False Negatives = 0

For correlation 4, the model yielded a prediction accuracy of 81.25%; the corresponding confusion matrix is shown in Fig.29.

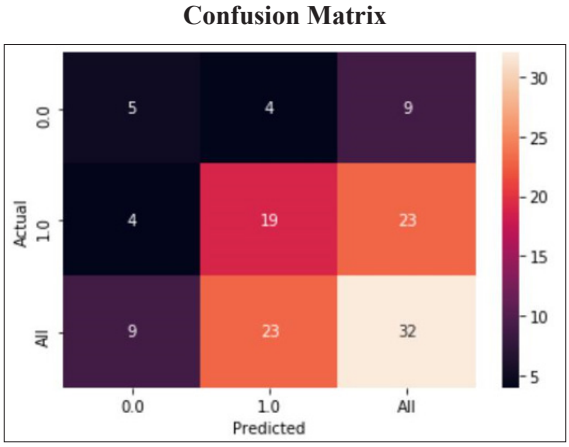

Figure 31: Confusion Matrix

TP = True Positives = 9
TN = True Negatives = 17
FP = False Positives = 3
FN = False Negatives = 3

Summary of Analysis
The summary of the analysis is captured as Table-1, wherein the respective correlations are plotted against their F1-scores

Table 1: Results

Model		Accuracy	Precision	Recall	F1 Score
Backpropagation	C1	0.94	0.95	0.95	0.95
	C2	0.91	0.87	1.00	0.93
	C3	0.97	0.96	1.00	0.98
	C4	0.91	0.90	0.94	0.92
Recurrent Neural Network (GRU)	C1	0.91	0.95	0.90	0.93
	C2	0.84	0.86	0.91	0.89
	C3	0.97	0.96	1.00	0.98
	C4	0.81	0.90	0.82	0.86
K Nearest Neighbor	C1	0.84	0.90	0.86	0.88
	C2	0.75	0.83	0.83	0.82
	C3	0.97	0.96	1.00	0.98
	C4	0.81	0.85	0.85	0.85

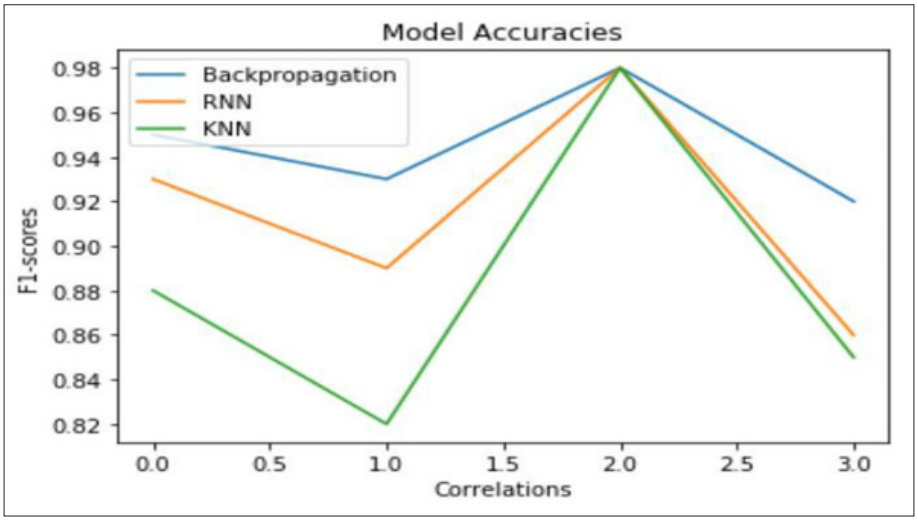

Figure 32: Model performance comparison graph based on F1-score

Conclusion

The objective of this study was to build a model that predicts Freshman student's aptitude for computer programming using Machine learning algorithms. Several hypotheses were conjectured, and corresponding questionnaire generated; they were given to school final students in India and the dataset was collected. Four models were built for each ANN based on four correlations generated using clustered hypotheses set; KNN was used as a classifier. The performance of the models was computed using the test dataset, which was 20 % of the original dataset. The results show that the BPN model/s achieved high accuracies in predicting the Freshman student's aptitude for computer programming. The best correlation scores for the clustered hypotheses C1, C2, C3 and C4 were 94%, 91%, 97%, 91% respectively. Although the best model was the BPN, it took the second longest time to train unlike the KNN, while the first being RNN. The results show that the models can be employed to predict Freshman student's aptitude for programming. Further work in this arena include the use of Convolutional Neural Network (CNN) to study student's aptitude for programming and also generate several useful 3-d metrics.

Brief Overview of the Algorithms Used

Multiple Linear Regression

Multi-Linear regression is the process of using many independent variables to determine one dependent variable (many to 1 relationship). In Multiple Linear Regression, we try to find relationship between two or more independent variables (inputs) and corresponding dependent variable (output). The independent variables can be continuous or categorical.

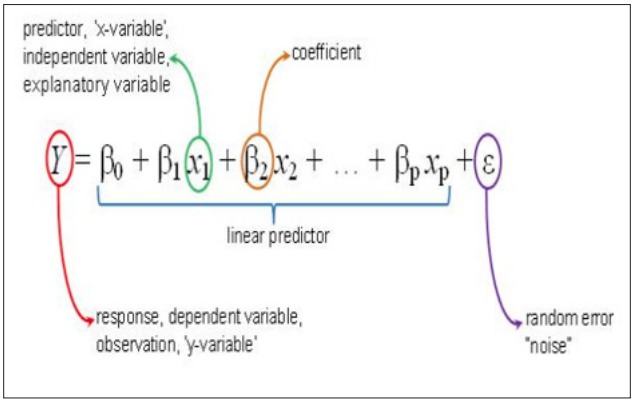

Figure 33: Multilinear Regression Formula

Artificial Neural Network (Ann)

Artificial Neural Network (ANN) can be defined as information processing tools which mimic or copy the learning methodology of the biological neural networks. It derives its origin from the human nervous system, which consists of massively parallel large interconnection of large number of neurons, which activate different perceptual and recognition task in small amount of time. The last part of the research dealt with focusing on the use of my ANN to come up with a model that would predict student aptitude for programming based on the hypothesis elucidated in 3-Parameter Classification. For these four different models based on the correlation from each of the hypothesis were designed. The selected artificial neural networks are:

Backpropagation Neural Network (BNN)

Backpropagation is a feed forward neural network algorithm, which works by computing the gradient of the loss function with respect to each weight by the chain rule, computing the gradient one layer at a time, iterating backward from the last layer to avoid redundant calculations of intermediate terms in the chain rule. Backpropagation is a short form for "backward propagation of errors." It is a standard method of training artificial neural networks. This method helps to calculate the gradient of a loss function with respect to all the weights in the network.

Recurrent Neural Network (RNN) – Gated Recurrent Network

Recurrent Neural Network (RNN) is a type of Neural Network where the output from previous step is fed as input to the current step. In traditional neural networks, all the inputs and outputs are independent of each other. Thus, RNN came into existence, which solved this issue with the help of a Hidden Layer. The main and most important feature of RNN is Hidden state, which remembers some information about a sequence. They are especially powerful in use cases in which context is critical to predicting an outcome and are distinct from other types of artificial neural networks because they use feedback loops to process a sequence of data that informs the final output, which can also be a sequence of data. The feedback loops allow information to persist; the effect is often described as memory. RNNs built with LSTM units categorize data into short term and long-term memory cells. Doing so enables RNNs to figure out data that is important and should be remembered and looped back into the network, and the data that can be forgotten or left out.

K-Nearest Neighbor

K-Nearest Neighbor (KNN) Algorithm uses the entire dataset in its training phase. Whenever a prediction is required for an unseen data instance, it searches through the entire training dataset for k-most similar instances and the data with the most similar instance is finally returned as the prediction. (Atul, 2020). Futhter, k-nearest neighbor algorithm uses a very simple approach to perform classification. When tested with a new example, it looks through the training data and finds the k-training examples that are closest to the new example.

Appendix-A: Questionnaire Employed

correlation	Hypothesis	Question	Mean	Mode	Median	Standard Deviation	Rho	Hypothesis Mean	Hypothesis STD
C1	H0	1. I have enough skills to learn programming.	3.8408	6	5	2.2773	0.235594	3.6943	2.3064
		2. I have enough knowledge to learn programming	3.5478	6	4	2.3355	0.265016		
	H1	6. Computer programming Is difficult for me	2.6752	1	2	2.1370	0.162884	2.6274	2.1621
		7. Computer programming is not for me	2.5796	1	1	2.1872	0.168882		
	H2	12. I know some programming and hence learning formal programming is easy	3.7898	6	5	2.3343	0.251579	3.3482	2.2713
		13. I can program faster, because of past experience in programming	3.4013	6	3	2.2613	0.241363		
		14. I do not like programming due to lack of experience with computer programming.	2.8535	1	2	2.2183	0.180883		
C2	H3	15. I have logical thinking skills and therefore programming is rather easy for me	3.9873	6	5	2.1122	0.262977	3.3217	2.0810
		16. I have difficulty in understanding logic and how it works and therefore I find programming difficult	2.6561	1	2	2.0497	0.141133		
	H5	20. I have the mental tenacity for handling difficult programming problems.	2.9618	6	3	2.2699	0.190287	3.0573	2.2857
		22. Mental tenacity has little relationship towards solving difficult programming problems	3.1529	6	3	2.3015	0.190886		
	H6	23. Visual tools help me well in learning programming	3.8344	6	5	2.3283	0.274012	3.0191	2.2104
		24. I prefer pseudo code tools for learning programming	2.7006	0	2	2.3300	0.234949		
		25. Visual tools are not helpful in learning/ understanding of Programming	2.5223	1	2	1.9728	0.219233		
C3	H8.0.	29. Gender of a person plays a role in learning programming	2.2611	1	1	2.0790	0.162337	2.1943	2.0799
		30. here exists gender bias in the process of learning	2.1274	1	1	2.0808	0.198905		

	H8.1.	31. I learn programming better through visual environments.	4.5159	6	6	2.4823	0.196949	4.1019	2.3727
		32. I learn programming better through collaborative learning environments	3.8471	6	5	2.3593	0.185174		
		33. I prefer Visual environment for learning programming.	3.9427	6	5	2.2765	0.264137		
	H8.2	34. Female students learn programming better through visual environments	2.8790	6	3	2.3543	0.157574	2.8599	2.3600
		35. Female students learn programming better through collaborative learning environments.	2.8408	6	3	2.3656	0.192149		
C4	H9	37. Learning programming calls for minimum level of logical Skills.	2.6752	0	2	2.1874	0.191117	2.8981	2.2249
		39. learning programming calls for minimum level of logical skills.	2.8981	6	3	2.1815	0.224642		
		40. I have good degree of logical skills	3.3503	6	4	2.2528	0.203063		
		41. I have good degree of experience in programming	2.5796	1	2	2.1547	0.137835		
		42. I have good degree of mental tenacity	2.9873	6	3	2.3479	0.150306		
	H10	44. Learning programming calls for a cumulative minimum level in the sum of Logical Skills and Mental Tenacity. Below the cumulative levels, students get disinterested in learning computer programming	2.5669	0	2	2.2454	0.101654	2.5669	2.2454
	H7	26. Collaborative learning environment helps me well in learning programming	3.9427	6	5	2.4370	0.222606	3.4055	2.3572
		27. I prefer collaborative learning environment for learning programming	6	6	5	2.3769	0.20388		
		28. collaborative learning environments are not helpful in learning/ understanding programming	2.5223	0	2	2.2577	0.162329		

References

1. Ahadi A, Lister R, Arto Vihavainen, Heikki Haapala (2015) Exploring Machine Learning Methods to Automatically Identify Students in Need of Assistance. Proceedings of the eleventh annual International Conference on International Computing Education Research - ICER 15 121-130.
2. Devasia M, Vinushree H, Vinayak Hegde (2016) Prediction of students performance using educational data mining. International conference on data mining and advanced computing (SAPIENCE) https://www.semanticscholar.org/paper/Prediction-of-students-performance-using-Data-Devasia-Vinushree/f5c308309f207c7aed7e1e7a8b7168d86f7c394f.
3. Longi K (2016) Exploring factors that affect performance on introductory programming courses 1-70 https://core.ac.uk/download/pdf/78562457.pdf.
4. Östblom J (2019) About us: Amethix Technologies. Retrieved from Amethix Technologies web site: https://amethix.com/waterfall-or-agile-the-best-methodology-for-ai-and-machine-learning/.
5. Ward M, Peters G, Shelley K (2010) Student and faculty perceptions of the quality of online learning experiences. Int Rev Res Open Distrib Learn 11: 57-77.
6. V Lakshmi Narasimhan (2019) Hypotheses Development and Survey Questionnaire for the Understanding of Freshman Aptitude for Programming, Unpublished Manuscript.
7. V Lakshmi Narasimhan (2021) Proactive Personalized Primary Care Information System- P3CIS. Information Science and Applications 739: 357:365.
8. V Sampath Kumar and V Lakshmi Narasimhan (2021) Using Deep Learning for Assessing Cybersecurity Economic Risks in Virtual Power Plants. IEEE Intl. Conf. on Electrical Energy Systems (ICEES) https://ieeexplore.ieee.org/document/9383723/references#references.
9. V Lakshmi Narasimhan (2020) Internet of Medical Things (IoMT). Proc of AICTE-ATAL Academy (International), University Grants Commission (UGC), India, (21-25, September 2020).
10. Diederik P Kingma, Jimmy Ba (2017) Adam: A Method for Stochastic Optimization. Computer Science https://arxiv.org/abs/1412.6980.
11. Weibo Liu, Zidong Wang, Yuan Yuan, Nianyin Zeng, Kate Hone, Xiaohui Liu (2019) A Novel Sigmoid-Function-Based Adaptive Weighted Particle Swarm Optimizer. IEEE Trans On Cybernetics 51: 1085-1093.