International Journal of
INTELLIGENT SYSTEMS AND APPLICATIONS IN ENGINEERING
ISSN:2147-6799 www.ijisae.org Original Research Paper

Deep Seated Neural Network Learner Model for Trust Recommendation

Jayashree V. Agarkhed[1*], Geetha Pawar[2]

Submitted: 24/04/2023 **Revised**: 26/06/2023 **Accepted**: 06/07/2023

Abstract: Recent and cutting-edge paradigms now used in the field of computers is pervasive computing. In comparison to traditional computer environments, its capacity to distribute computational services within settings where people live, work, or socialise makes problems like privacy, trust, and identification more difficult. One of the main issues with pervasive computing is the breach of security and privacy caused by malicious nodes. This study presents the new deep seated neural network leaner which recommends the trustworthy and unworthy transaction context in pervasive computing environment. Several models such as gaussian naive bayes, random forest, extra tree classifier, passive aggressive classifier, support vector machine learning algorithms are built. The proposed model out performs better compare to other machine learning models.

Keywords**:** *Pervasive Computing, Trust Recommendation, Machine learning, Deep neural network, Optimization*

1. Introduction

Information and communication technology (ICT) systems are evolving at an ever-increasing rate, going beyond large desktop computers to smaller and more potent devices that offer advanced processing capabilities and a variety of heterogeneous wireless communication interfaces. The fundamental benefit of pervasive computing environments is to improve quality of life by providing mobile devices and digital infrastructures that can distribute any kind of service in settings where people live, work, or socialize. However, pervasive computing also poses a number of security-related hazards and problems that raise a number of unresolved challenges. The design is made more difficult by the fact that pervasive systems are frequently invisible or embedded and take part in delivering the required service without the user's explicit or conscious understanding, either working or interacting with others [1]. Additionally, numerous devices operating in pervasive settings must undertake cross-device interactions without prior knowledge of one another and by fully autonomously differentiating themselves from one another. Therefore, it can be challenging for users to transmit personal information, such as identities, preferences, roles, and current positions, when these gadgets are present. The connection between numerous unknown entities creates a new issue in the aforementioned situations, mostly related to security management [2]. The security issues in pervasive computing are shown in Figure 1.

The introduction of adversarial and malicious entities poses significant risks in an operational environment, where the interacting entities are unable to learn anything about one another 's reliability before forming links. In such a situation, both service requesters and providers must continuously protect themselves from potential hostile user attacks, either by imposing certain communication restrictions or by taking advantage of some intricate and difficult security strategies/mechanisms. Therefore, a crucial problem in pervasive scenarios is the ability to autonomously discern between the trustworthy and untrustworthy entities involved in an encounter. Trust-based models are frequently used to assess the reliability of the interacting entities in order to address this issue [3].

Pervasive computing, as is more widely known, is a result of recent technological advancements in computing and communication that have multiplied the number of devices with embedded computational capabilities. Such devices continuously track people and their surroundings and gather a large amount of behavioral data [4]. Continuous video feeds of users' surroundings are crucial to emerging technologies, such as augmented reality [5]. Systems using artificial intelligence (AI) can use these data to discover more consumers. In addition to monitoring, advertising, and the algorithmizing of behaviour, this data gathering and analysis may be performed with the intention of improving service quality.

*1*Professor, Department of Computer Science and Engineering, Poojya Doddappa Appa College of Engineering, Kalaburagi, Karnataka, India*
*1*Corresponding Author Email: jayashreeptl@yahoo.com*
2Research Scholar, Department of Computer Science and Engineering, Poojya Doddappa Appa College of Engineering, Kalaburagi, Karnataka, India
2Email: geetamnayak@gmail.com

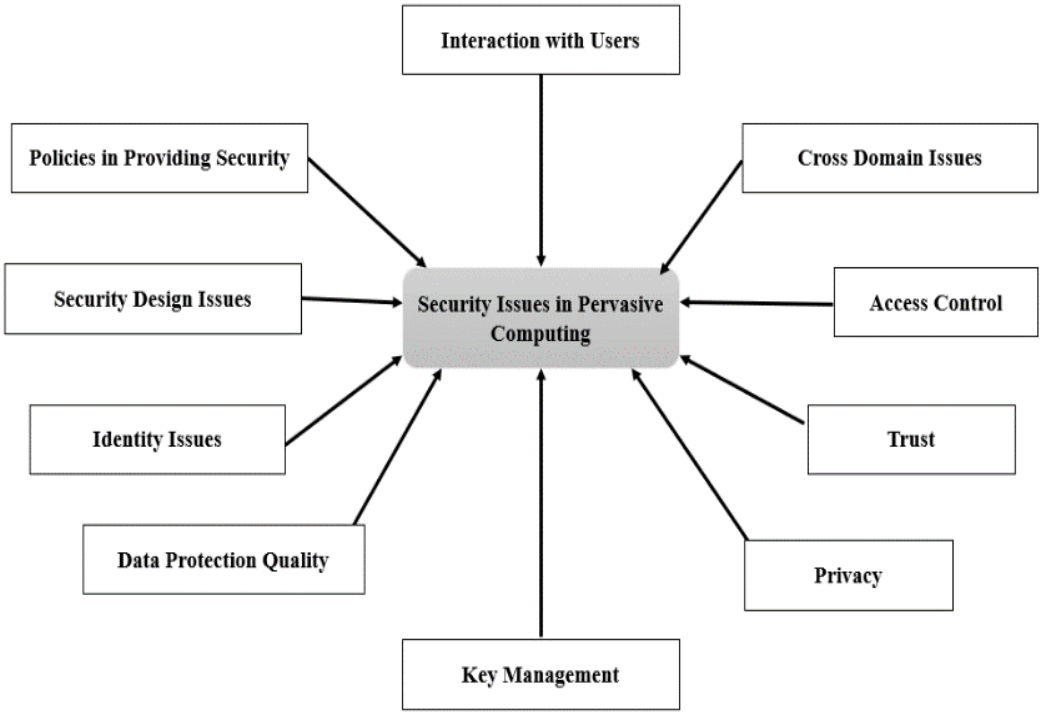


Fig 1: Security and Trust Issues in Pervasive Computing

In this work, the contribution is made by developing an efficient customized deep neural network, stacking deep learning models to ensure security concerns and trust over the pervasive computing. Different attributes in data set are analyzed using feature importance technique. The proposed model gives 100% accuracy and comparative study is carried out. The work presented in paper as follows. The background and associated works are covered in Section 2. The work flow and techniques employed in this research are described in Section 3. The results and implementation discussions are provided in Section 4. Finally, the research's conclusions are presented in Section 5.

2. Related Work

Since the 1990s, the security and trust pervasive computing research community has emphasized trust management model building and proposed many reliable approaches. The first decentralized trust-management Policymaker was proposed by Blaze et al. [6]. They used secure application policies and credential verification as the foundation of their trust architecture to control access to resources and services. It did not, however, employ recommendations to select a reliable, suitable provider. Additionally, due to its complicated processing requirements, it was not practical for a ubiquitous environment. According to Kagal et al. [7, 8], centralised security solutions do not scale effectively in large, open systems. As an alternative, they provide a security solution (Centaurus) based on trust management, which entails creating a security policy and giving entities access credentials. Centaurus significantly relies on the transfer of trust to third parties. For P2P environments, Damiani et al. [9] presented a protocol. The involved peers can maintain tabs on information about their fellow participants and can disseminate it throughout the group. The reliability of the other peers is evaluated by the peers using a distributed polling mechanism. However, only the Base Station attack is taken into account. A approach employing the entropy to assess how a rating differs from others was proposed by Weng et al. in [14]. The method starts the algorithm by using local ratings. Thus, if this initial rating differs significantly from previous ones, it might not be able to function properly. A formal trust model (FTM)-based control chart technique has been published in [15]. For P2P networks. Eigen Trust [16] is a well-liked reputation management tool. It is built on the idea of a friend circle, which makes it possible to assess an entity's reputation. The primary eigenvector of a matrix of local trust values is calculated to perform the trust evaluation. This method considers the entire history of contacts a peer has had with the system. If a collection of dishonest people creates a malevolent subnetwork of friends, the strategy fails. The technique suggested in [17] was called XRep. It is used to construct a self-policing P2P system capable of excluding all deemed unfair nodes from the network.

Pervasive computing trust management and security attacks detection have evolved over time towards the implementation of machine learning techniques in conventional Internet intrusion detection systems.

Wisanwanichthan and Thammawichai [10] employ SVM and Naive Bayes methods for feature reduction and normalization in their application of machine learning to intrusion detection systems (IDS). The main drawback of the machine learning-based intrusion detection system is that it takes a long time to train because it needs to process several datasets of prior data streams in the network. Dey [11], proposed a model by combining convolutional neural network and long short term machine model to analyses the NSL-KDD dataset [11]. A hybrid deep neural network (DNN) model was created by Chen et al. [12] for categorizing and identifying unknown network threats. According to Chen et al. [12], deep learning has attracted a lot of interest lately. They contrasted traditional approaches with the newest deep learning techniques. With the aid of deep learning's intelligent capabilities, Chen et al. [13] created an intelligent intrusion detection system.

3. Work Flow in the Study

To carry out the study systematically, the work flow was formed. It consists of data acquisition from dishonest internet user dataset, analyzing using data descriptive statistics, exploratory data analysis, pre-processing of data, model development, applying hyper parameter tuning, finding the best robust model and recommendation of trustworthy or untrustworthy. The data was developed using three different assault techniques. Attacks that are counted, timed, or context-based help the user build a reputation. By observing how users interact with one another on the network, the proposed approach makes it possible to assess each user's trustworthiness [18, 19]. The proposed work flow diagram of the study is shown in Figure 2. Table 1 shows that the Dishonest internet user dataset description.

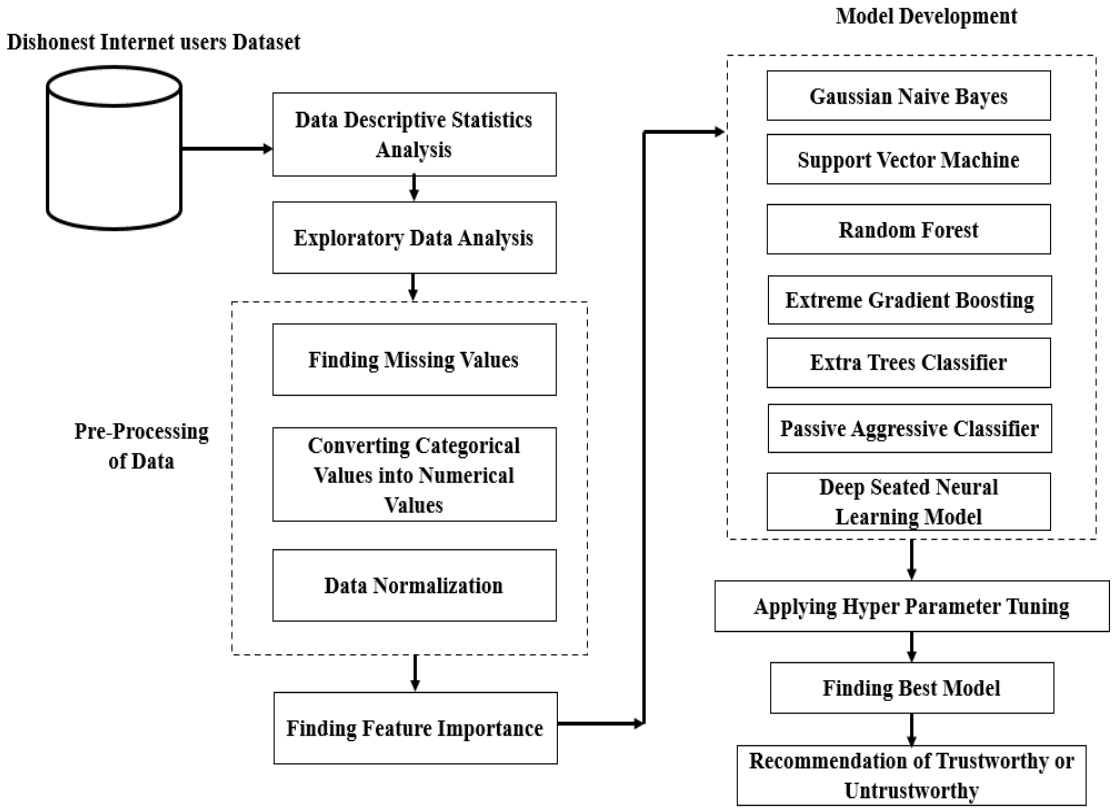


Fig 2: Work flow diagram for the study**.**

Table 1: Dishonest internet user dataset description

Attribute Name	Attribute Abbreviation	Description
Counting Trust	CT	It is employed to determine the number of reliable transactions that follow the final unreliable transaction (belonging to a certain context).
Counting of Untrust	CU	It is used to determine how many not trustworthy transactions (belonging to a particular context) follow the last reliable transaction.
Last Time	LT	It is employed to account for the time at which the most recent experience in a

		certain situation occurred.
Transactions Context	TC	It is used to specify the sort of transaction, such as a game, an online purchase, a social network, and others.
Trust Score	TS	It is the rating that one entity assigns to another at the conclusion of every direct interaction.

3.1 Data Descriptive Analysis

The interacting users of pervasive computing are unable to discover each other's level of trustworthiness. Because of this, malicious users may behave unfairly towards others. By observing how users interact with one another on the network, the suggested technique makes it possible to assess each user's trustworthiness. Tuples with important parameters that represent these behaviour are called tuples.

The design integrates some artificial intelligence-based technologies based on these tuples to construct a decision-making system. The descriptive analysis carried out is shown in table 2, table 3 shows the count of transaction, such as a game, an online purchase, a social network, and others. From table it shows that there are 322 records in the dataset and mean value is 2.195 in CT (ctrust), 1.5279 CU (cuntrust) and LT (last), similarly mean, standard deviation, minimum and maximum values is each column is represented.

Table 2: Data descriptive statistics analysis

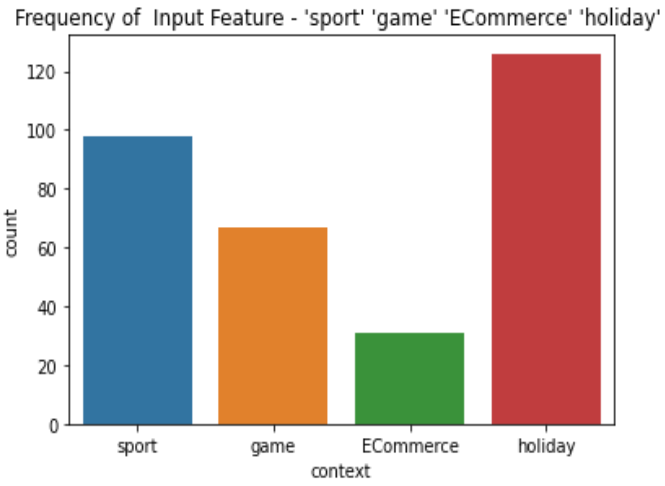
	ctrust	cuntrust	last
count	322.000000	322.000000	322.000000
mean	2.195652	1.527950	2.366460
std	1.273592	1.105422	1.347384
min	1.000000	1.000000	1.000000
25%	1.000000	1.000000	1.000000
50%	2.000000	1.000000	2.000000
75%	4.000000	1.000000	4.000000
max	4.000000	5.000000	4.000000

Table 3: Statistics of transaction context column

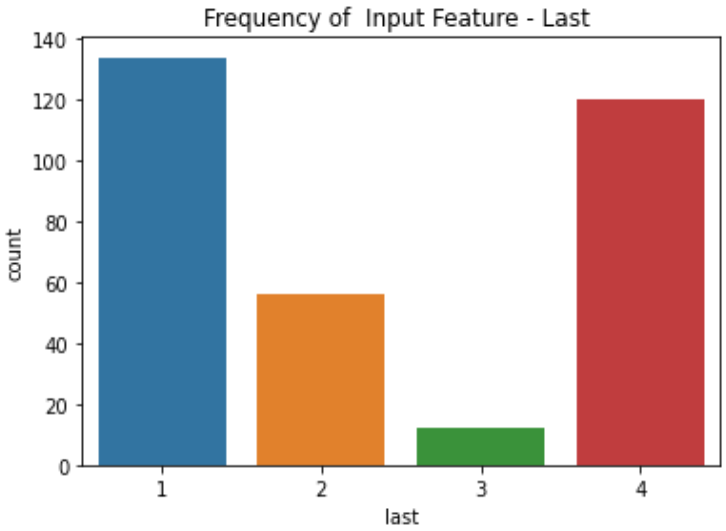
	Context	Counts
0	holiday	126
1	sport	98
2	game	67
3	ECommerce	31

3.2 Exploratory Data Analysis

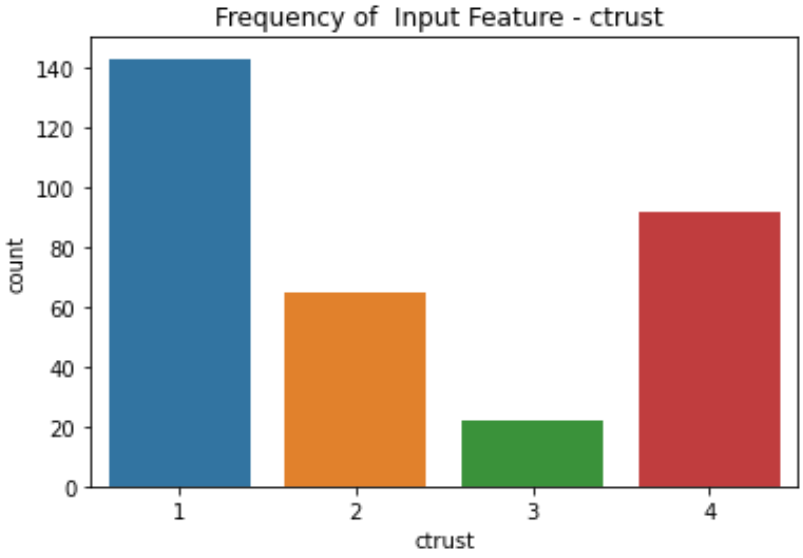
Exploratory data analysis is carried out to understand the insights of the data set. Frequency of each attributes is represented in bar graph as shown in Figure 3. Figure 4 represents distribution of trustworthy and untrustworthy transactions. It consists of 225 trustworthy transactions and 97 untrustworthy transactions. From we came to know that data is imbalance and skewed. Correlation of attributes such as CT, CU, LT, are studied to understand the relationship between the variable. Statistical techniques like correlation matrixes can be helpful when evaluating relationships between two or more variables in a data set. Figure 5 shows the correlation of attributes. Every column in the matrix has a correlation coefficient, which is a table where 1 indicates a strong association between variables, 0 indicates a neutral relationship, and -1 indicates a weak relationship.


Frequency of Input Feature - 'sport' 'game' 'ECommerce' 'holiday'
count
context
sport
game
ECommerce
holiday
0
20
40
60
80
100
120

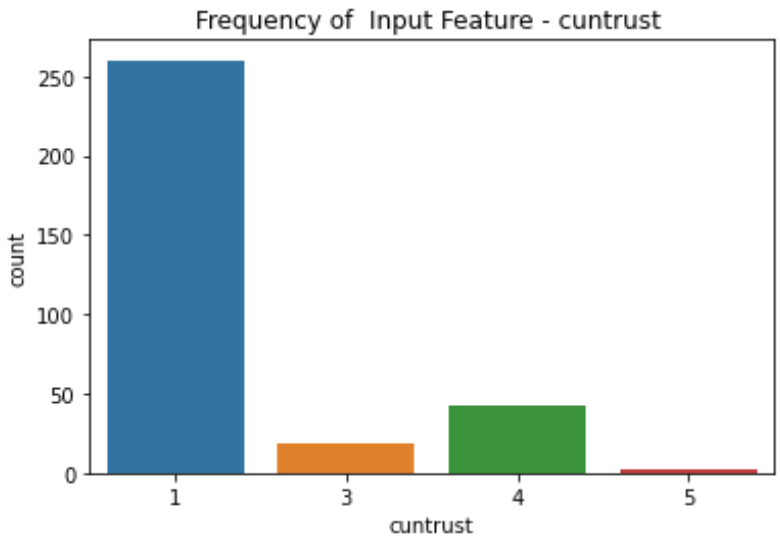
(a)


Frequency of Input Feature - Last
count
last
1
2
3
4
0
20
40
60
80
100
120
140

(b)


Frequency of Input Feature - ctrust
count
ctrust
1
2
3
4
0
20
40
60
80
100
120
140

(c)



(d)

Fig 3: Frequency of input features (a) transaction context (b) LT (c) CT (d) CU

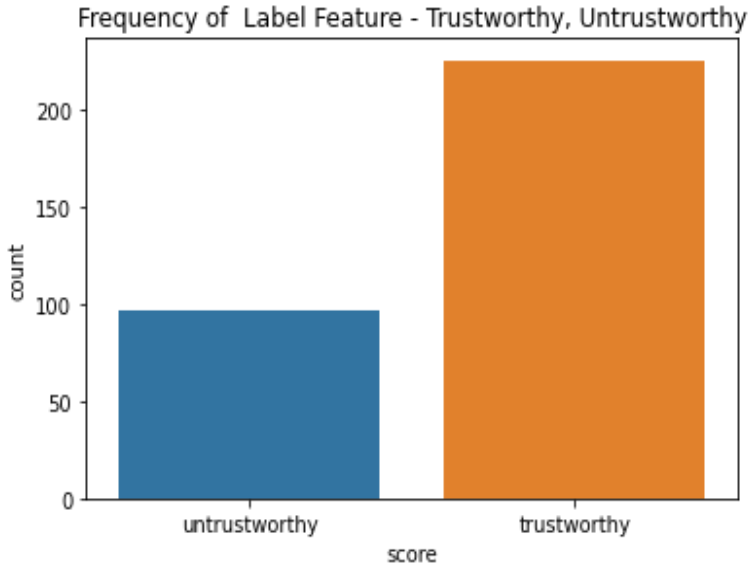


Fig 4: Distribution of trustworthy and untrustworthy transactions

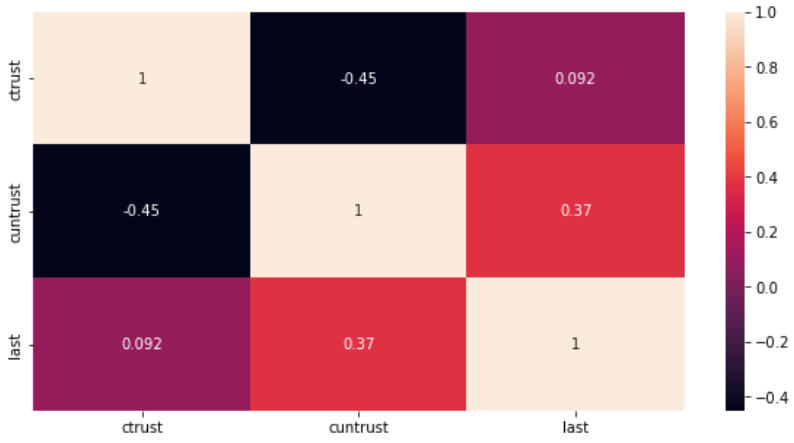


Fig 5: Correlation graph

3.3 Pre-Processing of Data

The Data Pre-processing stage includes Feature Scaling. The column TC the categorical values are converted into numerical values. Assigned 'sport':1,'game':2,'ECommerce':3,'holiday':4 and for score 'untrustworthy':0,'trustworthy':1. Next step is feature scaling, it is a technique for standardizing the variety of features in data that is also referred to as normalization. Feature Scaling is crucial since the input variables' scales for the data can differ. There are numerous scalars available in Python's sklearn module, including the MinMax Scaler, Standard Scaler, and Robust Scalar. One of the most well-liked scaling algorithms is the MinMax Scalar. By scaling each feature to a predetermined range, often [0,1] or [-1,1] in the event of negative values, it converts the characteristics. The MinMax Scalar applies the formula to each feature.

$$X' = \frac{X - X_{\text{minimum}}}{X_{\text{maximum}} - X_{\text{minimum}}} \quad (1)$$

It divides by the range, max(x)-min(x), after removing the column's mean from each value. When the standard deviation is very tiny or when there is no Gaussian distribution, this scaling approach performs very well. To the column CT, CU, LT and TC minmax scalar are applied for data normalization. In order to find the importance of the each feature the feature importance evaluation is carried using extreme gradient boosting technique. The Figure 6 shows the distribution of feature score of each attribute. From this graphs its shows that TC is having the highest feature importance score of 46.791, next highest is LT and CT is having the low feature importance score.

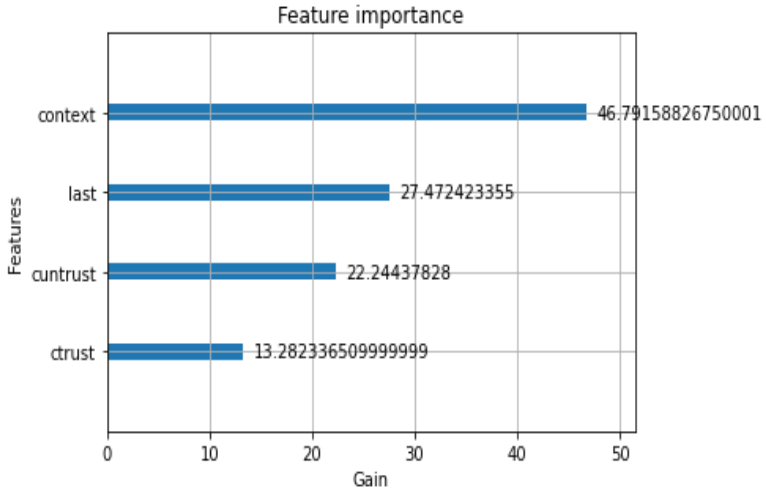


Fig 6: Feature important of each attribute

3.4 Model Development

To build the efficient recommendation of trust model, we tried with many machine learning and deep neural network models. The subsection explains all the model's description.

3.4.1 Gaussian Naïve bayes

It is an algorithm that discovers each object's probability, its characteristics, and the groups to which they belong. It also goes by the name "probabilistic classifier." The Naive Bayes method is built on the concept of probability. This algorithm's design is centred on the probabilistic solutions it can provide for issues that cannot be solved through prediction. The Bayes Theory relies on formulating a hypothesis (H) based on the available evidence (E). It has to do with two things: the likelihood of the hypothesis prior to the evidence P(H) and the likelihood of the hypothesis following the evidence P(H|E). When the qualities are continuous, gaussian naïve bayes algorithm to utilise. The attributes found in the data should adhere to the normal or gaussian distribution criterion. It significantly speeds up the search, but under lenient circumstances, the error will be twice as large as with Optimal Naive Bayes. The naïve bayes formula is as follows:

$$P(H/E) = (P(E/H) * P(H)) / P(\text{E}) \text{ ----} \quad (2)$$

where,

P(H|E) represents the behaviour of event H in the presence of event E.

P(E|H) denotes the frequency with which event E occurs when event H occurs first.

P(H) denotes the likelihood that event X will occur on its own.

The likelihood that event Y will occur on its own is represented by P(E).

3.4.2 Support Vector Machine

Making a straight line between two classes is how a straightforward linear SVM classifier functions. In other words, the data points on one side of the line will all be assigned to one category, while the data points on the other side of the line will be assigned to a different category. An SVM model is just a hyperplane in multidimensional space that represents several classes. SVM will generate the hyperplane in an iterative manner in order to reduce error. The kernel used to implement the SVM algorithm changes the input data space into the desired format. The kernel technique is a method used by SVM, in which the kernel converts a low-dimensional input space into a higher-dimensional space. The SVM linear kernel function is given by

$$f(x) = w^T * X + b \text{ ----} \quad (3)$$

In this equation, w stands for the weight vector which needs to be minimise, X for the data to classify, and b for the predicted linear coefficient from the training set. The decision boundary that the SVM returns is defined by this equation.

3.4.3 Random Forest

The structure of random forests (RF) is created by combining many classifier decision trees. Each decision tree is trained by randomly choosing features, and then the trees are built using the optimal split values and entropy. The major challenging task is that each decision tree will represent characteristics differently, which may have an impact on the classifier's performance. Co-relation issues between decision trees that on average produce Gaussian distributions are possible. Each decision tree's average value will be taken into account for classification. Figure 7 shows the RF structure.

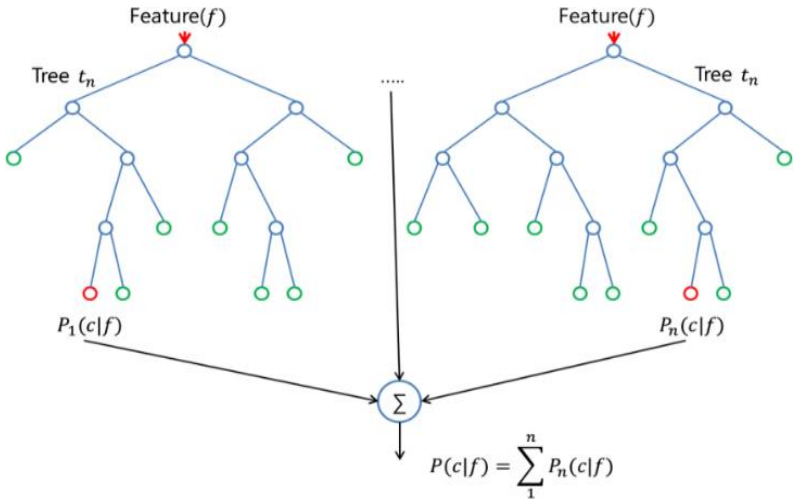


Fig 7: Structure of Random Forest Model

3.4.4 Extreme Grading Boosting

eXtreme Gradient Boosting (XGBoost) is a decision tree-based ensemble machine learning system that employs gradient boosting 20. Machine learning ensembles integrate the prediction results of various learnt models. The aggregated models may have been created using the same process or a different algorithm. The most widely used ensemble learning approaches are boosting and bagging. Many decision trees are generated in parallel from the initial learners in the bagging process.

3.4.5 Extra Trees Classifier

Extra Trees (ET) is an ensemble machine learning method that creates predictions by combining the output of a group of decision trees that have all been trained. To ensure that the decision trees are sufficiently varied, Random Forest employs bagging to choose various versions of the training data. However, Extra Trees trains decision trees using the complete dataset. Using the entire dataset allows Extra Trees to reduce the bias of the model. However, the randomization of the feature value at which to split, increases the bias and variance

3.4.6 Passive Aggressive classifier

The passive aggressive (PA) classifier algorithm, which belongs to the class of online learning algorithms, is capable of handling enormous datasets and alters its model in response to each new instance it meets. The passive aggressive algorithm can modify its weights when new information is received because it is an online learning algorithm. The regularisation parameter, C, of the passive aggressive classifier enables a trade-off between the margin's size and the amount of misclassifications. The passive aggressive classifier examines a new instance at each iteration, determines whether it was correctly

classified or not, and then modifies its weights in accordance. There is no change in weight if the case is correctly categorised. In contrast, if it is wrongly classified, the passive aggressive algorithm modifies its weights based on this incorrectly classified instance to better classify subsequent occurrences. The regularisation parameter C and the level of confidence in the categorization of that specific instance determine how much the Passive Aggressive algorithm updates its weights.

3.4.7 Deep Seated Neural Network Learner Model

For the purpose of classifying trustworthy and untrustworthy, we developed a new Deep Seated Neural Network Learner (DSNNL) model architecture. The DSNNL model architecture is shown in Figure 8. Multiple layers make up a neural network, and each layer is in charge of a distinct function. A neural network's layer count increases in direct proportion to how complex the underlying system is. This kind of model pushes the inputted data into a number of layers after receiving it.

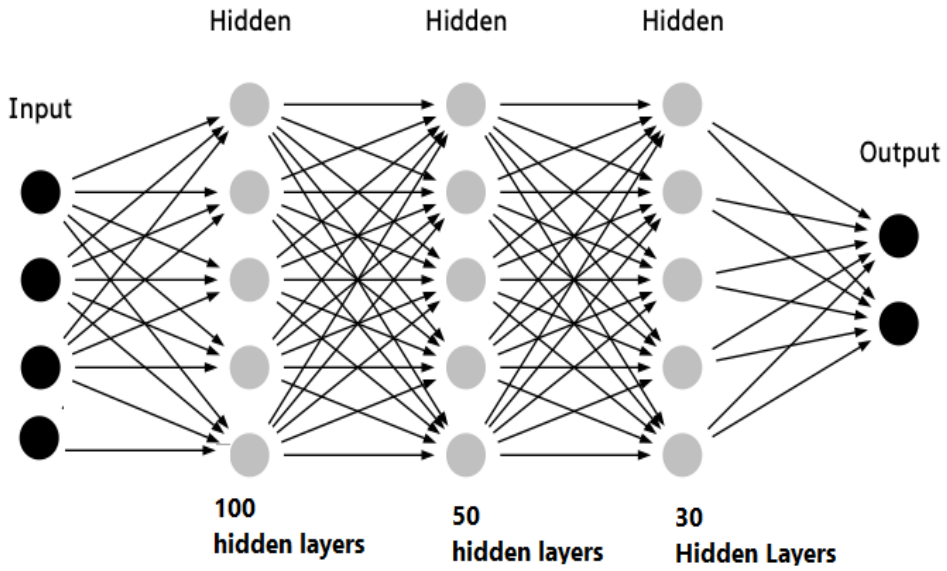


Fig 8: Deep Seated Neural Network Learner (DSNNL) model architecture

Input layers use actual data as their input, therefore they receive actual values from the data. Between the input and output layers are the following layers, known as the hidden layers. It is regarded as a deep neural network if it has three or more hidden layers. Activation functions are the method used in the various non-linear transformations that make up deep learning. As a result, activation functions are used prior to applying an input signal to the neural network's next layer. A neural network may learn complex attributes to activation functions. In deep neural network there are many activation functions such as threshold, sigmoid, rectifier (relu), hyperbolic tangent and cost function. Initial design of the model we used 'relu' activation function. One of the most used functions in the area of deep learning is the Rectifier Linear (ReLU) function. The value of x is first set to 0 if the value is less than or equal to 0, after which it steadily advances as the input value rises. In the second and third hidden layers we have used sigmoid activation function. The equation for sigmoid function is as follows:

$$y = \frac{1}{1 + e^{-x}} \text{ .---} \quad (4)$$

A logistic function is utilised in the sigmoid function. Anything below 0 in this method will be set to 0. In the output layer, this function is frequently utilised, especially when attempting to determine the probability of predictive values. The proposed DSNNL algorithm is explained below:

Proposed DSNNL Algorithm

Let the dataset $S = \{(\mathrm{F}_i, \mathrm{T}_j) \mid 1 \le i \le N, 1 \le i \le M\}$, where N is the size of the training data.

The input features $\mathrm{F}_i = \{\mathrm{f}_1, f_2, \ldots\ldots\ldots f_n\}$ be the n features and T_j be the label for the input features F_i

Compute output of each neuron $z = \sum w_i * x_i + b$ where w_i is the weights and b represent the bias and compute the a value using z which is equal to the output layer $a = \varphi(z)$ which is the activation function.

Initialise the model hyper parameters:

- Learning rate of the model = 0.01, 0.001, 0.0001
- Number of epochs = 50, 100, 200
- Select the best activation function:ReLu, Sigmoid, Tanh, LeakReLu
- Number of layers number of units in each neural network layer
- Optimiser: adam, adamw, RMSprop

Initialise the accuracy value best = 0.0

Create_best_dsnnl_model(test_data)

for model in dsnnl_model_hyperparametes

do

for i= 1,2,…… N: units in each hidden layer

do

perform forward propagation $\forall_i$

compute predict the output y_i evaluate the function

$$K(\theta) = \frac{1}{p} \sum_{i=1}^{p} C(y_i^{\theta}, \mathrm{y}_i) \text{ -----(5)}$$

Execute back propagation

$$\mathrm{f(x, y, z)} = T\left[\frac{df}{dx} \ \frac{df}{dy} \ \frac{df}{dz} \right] \text{---- (6)}$$

$\theta =: G(\theta)$ and compute accuracy

if accuracy is best

best = accuracy

done

done

return the best_dsnnl_model

4. Implementation and Results

In this work to conduct the experiment and implement the models we have built with GPU enabled system on a 64-bit Windows computer with 25.5 GB of RAM and a graphics processor unit with 15 gigabytes of system memory, Google's Collaboratory Pro environment was used to conduct all experiments. The greatest amount of storage space offered was 166.8 GB. All the scripts are implemented using the Keras framework, a Python toolkit for deep neural networks that is free and open-source. Figure 9 displays an example of a hardware system specification that is used in the backend. Table 4 shows the parameters and corresponding values for XGBoost classifier here.

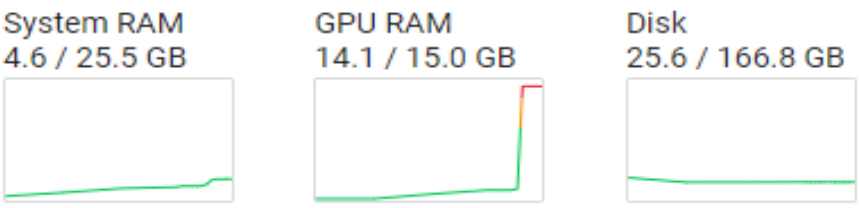


Fig 9. Hardware System Specifications in the Backend

Table 4. Parameter and Values set for XGBoost Classifier

Parameter	Value
random_seed	42
n_estimators	10
n_jobs	-1
max_depth	6
scoring	'f1'

The DSNNL model is built sequentially. The sequential model looks similar to a layer-by-layer linear stack. It is helpful for creating basic models like encoder-decoder models and the simple categorization network. Essentially, it treats the layer as a feed for the following layer. The model is built as a sequential model, and then a flatten layer will be added. We have 100 neurons in the hidden layer 1, 50 neurons in the hidden layer 2, and 30 neurons in the hidden layer 3. The Rectified Linear Units (ReLU) activation function. Due to its single neuron count and sigmoidal function, the last layer converts logits into probabilities that add up to one. The Adam optimiser is used to optimise the model. Adam is built using a first-order gradient optimizer that makes estimates using lower-order moments. This takes up less memory, is invariant to diagonal rescaling, and works with big input parameters. Adam works well for non-stationary, noisy, and sparse gradient targets. Training and validation loss are estimated to assess the models developed. The validation loss is assessed after each epoch while the training loss is calculated after each batch execution. Earlier, an average training loss was computed per epoch. In terms of evaluating validation loss, it offers advantages over gradient updates. To minimise the loss function while training, the model quantity must be found. To determine the loss in predicted and true classes, binary cross entropy is computed in this situation. When the batch size as a whole is used, the loss will provide the average per-sample loss for that batch. The accurately predicted values on the test dataset are added to the total number of input datasets to calculate the model accuracy. Table 5 gives the equation for calculating model correctness. Table 6 shows the obtained model accuracy and comparison of different model build using evaluation metrics precision, recall, f1-score and accuracy. The proposed model DSNNL out performs compare to other state-of-the-art models. Figure 10 shows the accuracy comparison, the DSNNL model gives the highest score of 100% accuracy

Table 5: Evaluation metrics used

Measures	Formula	Description
Recall (R)	*True positive (TP) /(True Positive (TP)+ False Negative (FN))*	Count of faulty modules rightly classified
Precision (P)	*True positive (TP)/(True Positive (TP)+ False Positive (FP))*	Count of modules rightly classified as faulty
F1-score	*2 *((P*R)/(P+R)*	It combines a model's precision and recall scores.
Accuracy	*(TP+TN)/(TP+FP+TN+FN)*	Accuracy of the model

Table 6: Comparison of the build model precision, recall, f1-score and accuracy

Build Models	Class	Precision	Recall	F1-score	Accuracy (%)
Gaussian NB	0	0.56	0.37	0.45	72.84
	1	0.76	0.87	0.81	
SVM	0	0.75	0.62	0.68	82.72
	1	0.85	0.91	0.88	
RF	0	0.90	0.95	0.92	89.55
	1	0.88	0.79	0.83	
XGBoost	0	1.00	0.98	0.99	98.45
	1	0.96	0.99	0.97	
ET	0	0.99	0.97	0.97	97.65
	1	0.92	0.97	0.94	
PA	0	0.93	0.62	0.75	87.65
	1	0.86	0.98	0.91	
DSNNL (Proposed)	0	1.00	1.00	1.00	100
	1	1.00	1.00	1.00	

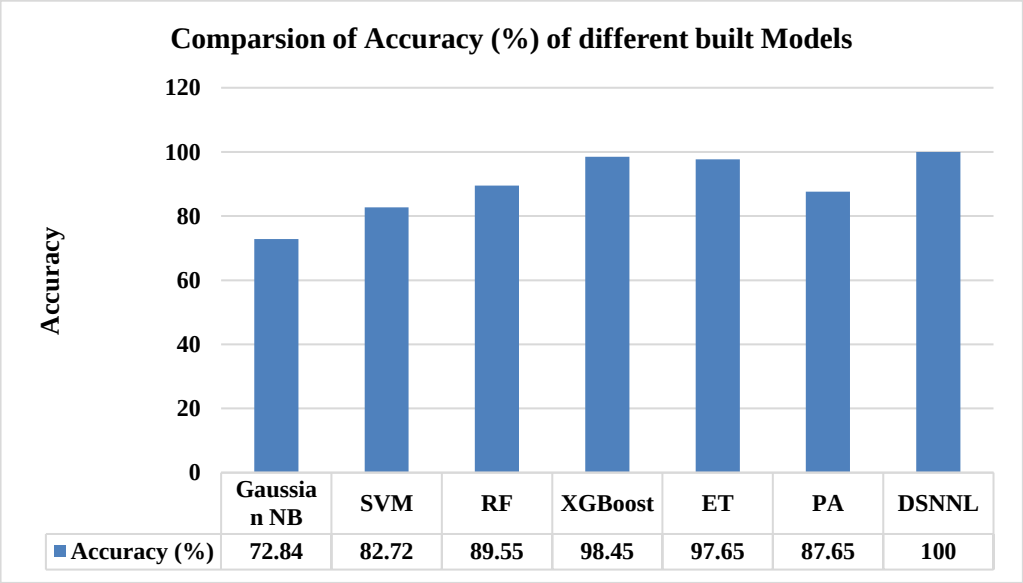


Fig 10: Comparison of accuracy of various build models.

5. Conclusion

In this study a new deep seated neural network model is created to recommend the trust in pervasive computing environment. The dishonest internet user dataset is analysed using descriptive statistical analysis and exploratory data analysis techniques. To normalise the data min-max scaling technique is applied and converted categorical data into numerical data. To understand the features' important extreme gradient boosting method is utilised. Various machine learning models such as gaussian naive bayes, random forest, extra tree classifier, passive aggressive classifier, support vector machine is built and evaluated with measuring metrics. The proposed DSNNL model gives an accuracy score of 100 percent. The future work will be carried out the multiple source data and will try to fuse the dataset to understand the more insights and build more robust trust recommendation model in pervasive computing.

References

[1] F. Almenárez, A. Marín, D. Díaz, A. Cortés, C. Campo, and C. García-Rubio, 2011. Trust management for multimedia P2P applications in autonomic networking. *Ad Hoc Networks*, *9*(4), pp.687-697. doi:10.1016/j.adhoc.2010.09.005.

[2] N. Iltaf, A. Ghafoor, and M. Hussain, 2012. Modeling interaction using trust and recommendation in ubiquitous computing environment. *EURASIP Journal on Wireless Communications and Networking*, *2012*, pp.1-13.doi:10.1186/1687-1499-2012-119.

[3] T. Sun, M.K. Denko, 2007. A distributed trust management scheme in the pervasive computing environment, in: Proceedings of the Canadian Conference on Electrical and Computer Engineering, pp. 1219–1222. doi:10.1109/CCECE.2007.311.

[4] H. Haddadi, P. Hui, T. Henderson, and I. Brown, Targeted advertising on the handset: Privacy and security challenges," in Pervasive Advertising. Springer, 2011, pp. 119–137.DOIhttps://doi.org/10.1007/978-0-85729-352-7_6

[5] C. Bermejo, Z. Huang, T. Braud, and P. Hui, 2017. When augmented reality meets big data, in 2017 IEEE 37th International Conference on Distributed Computing Systems Workshops (ICDCSW), June pp. 169–174.DOI: 10.1109/ICDCSW.2017.62

[6] M Blaze, J Feigenbaum, J Lacy, 1996. Decentralized trust management, in 17th IEEE Symposium on Security and Privacy, *Oakland*, pp. 164–173.DOI: 10.1109/SECPRI.1996.502679

[7] L.Kagal, T.Finin, A. Joshi, 2001. trust-based security in pervasive computing environments. *IEEE Comput.* 34(12), pp.154–157.DOI: 10.1109/2.970591

[8] L. Kagal, F. Perich, A.Joshi, and T. Finin, 2002, October. A security architecture based on trust management for pervasive computing systems. In *Grace Hopper Celebration of Women in Computing*.

[9] E. Damiani, D.C. di Vimercati, S. Paraboschi, P.Samarati, and F. Violante, 2002, November. A reputation-based approach for choosing reliable resources in peer-to-peer networks. In *Proceedings of the 9th ACM conference on Computer and communications security* (pp. 207-216). doi:10.1145/511446.511496.

[10] T.Wisanwanichthan, and M. Thammawichai, 2021. A double-layered hybrid approach for network intrusion detection system using combined naive bayes and SVM. *IEEE Access*, *9*, pp.138432-138450.DOI: 10.1109/ACCESS.2021.3118573

[11] A. Dey, 2020, December. Deep IDS: A deep learning approach for Intrusion detection based on IDS 2018. In *2020 2nd International Conference on Sustainable Technologies for Industry 4.0 (STI)* (pp. 1-5). IEEE.DOI: 10.1109/STI50764.2020.9350411

[12] P. Chen, Y. Guo, J. Zhang, Y.Wang, and H. Hu, 2020, December. A novel preprocessing methodology for dnn-based intrusion detection. In *2020 IEEE 6th International Conference on Computer and Communications (ICCC)* (pp. 2059-2064). IEEE.DOI: 10.1109/ICCC51575.2020.9345300

[13] L. Chen, X. Kuang, A. Xu, S. Suo, and Y. Yang, 2020, December. A novel network intrusion detection system based on CNN. In *2020 eighth international conference on advanced cloud and big data (CBD)* (pp. 243-247). IEEE.DOI: 10.1109/CBD51900.2020.00051

[14] J. Weng, C.Miao, and A. Goh, 2005, August. Protecting online rating systems from unfair ratings. In *International Conference on Trust, Privacy and Security in Digital Business* (pp. 50-59). Berlin, Heidelberg: Springer Berlin Heidelberg.doi:10.1007/11537878_6.

[15] S.I. Ahamed, M.M. Haque, M.E. Hoque, F.Rahman, and N. Talukder, 2010. Design, analysis, and deployment of omnipresent formal trust model (FTM) with trust bootstrapping for pervasive environments. *Journal of Systems and Software*, *83*(2), pp.253-270. doi:10.1016/j.jss.2009.09.040.

[16] S.D. Kamvar, M.T.Schlosser, and H. Garcia-Molina, 2003, May. The eigentrust algorithm for reputation management in p2p networks. In *Proceedings of the 12th international conference on World Wide Web* (pp. 640-651). doi:10.1145/775152.775242.

[17] E. Damiani, D.C. di Vimercati, S. Paraboschi, P.Samarati, and F. Violante, 2002, November. A reputation-based approach for choosing reliable resources in peer-to-peer networks. In *Proceedings of the 9th ACM conference on Computer and communications security* (pp. 207-216). doi:10.1145/586110.586138.

[18] G. D'Angelo, S.Rampone, and F. Palmieri, 2017. Developing a trust model for pervasive computing based on Apriori association rules learning and Bayesian classification. *Soft Computing*, *21*, pp.6297-6315. DOI: 10.1007/s00500-016-2183-1

[19] G. DAngelo, S.Rampone, and F. Palmieri, 2015, November. An artificial intelligence-based trust model for pervasive computing. In 2015 10th international conference on P2p, parallel, grid, cloud and internet computing (3pgcic) (pp. 701-706). IEEE. DOI: 10.1109/3PGCIC.2015.94.

[20] Prof. Madhuri Zambre. (2016). Analysis and Modeling of Physical Stratum for Power Line Communication. International Journal of New Practices in Management and Engineering, 5(01), 08 - 13. Retrieved from

http://ijnpme.org/index.php/IJNPME/article/view/42

[21] Pareek, M., Gupta, S., Lanke, G. R., & Dhabliya, D. (2023). Anamoly Detection in Very Large Scale System using Big Data. SK Gupta, GR Lanke, M Pareek, M Mittal, D Dhabliya, T Venkatesh,.." Anamoly Detection in Very Large Scale System Using Big Data. 2022 International Conference on Knowledge Engineering and Communication Systems (ICKES).

[22] Mohapatra, S. K. ., Patnaik, S. ., & Kumar Mohapatra, S. . (2023). An Enhanced Automated Epileptic Seizure Detection Using ANFIS, FFA and EPSO Algorithms . International Journal on Recent and Innovation Trends in Computing and Communication, 11(4s), 57–67. https://doi.org/10.17762/ijritcc.v11i4s.6307