https://doi.org/10.33472/AFJBS.6.6.2024.6363-6380

African Journal of Biological Sciences

Journal homepage: http://www.afjbs.com

Research Paper Open Access

Performance Evaluation of Deep Learning Techniques in Distributed Computing and Traditional Computing Environments for Structured Data Analysis

M. Bhargavi Krishna[1], Prof. S. Jyothi[2], Dr. P. Bhargavi[3]

[1]Research Scholar, Dept. of. CSE, SOET, Sri Padmavathi Mahila Visvavidyalayam, Tirupati

[2] Professor, Dept. of. Computer Science, Sri Padmavathi Mahila Visvavidyalayam, Tirupati

[3]Assistant Professor, Dept. of. Computer Science, Sri Padmavathi Mahila Visvavidyalayam, Tirupati

Article Info

Volume 6, Issue 6, June 2024

Received: 26 April 2024

Accepted: 02 June 2024

Published: 28 June 2024

doi: 10.33472/AFJBS.6.6.2024.6363-6380

ABSTRACT:

Handling large-scale data is a formidable task for data scientists and researchers due to the rapid generation and ever-growing volume of data. The sources encompass a wide range of current datasets and databases that contain both structured and unstructured data. Hence, the utilisation of advanced algorithms is important to effectively target the vast amount of data before it becomes inaccessible to the current algorithms. Distributed computing has gained popularity as it provides superior scalability and performance compared to Traditional Computing Systems. This paper analyses the structured data with the latest algorithms like Random Forest Classifier, Extra Trees Classifier, Gradient Boosting Regressor, Gradient Boosting Classifier, voting classifier, Federated Learning, Distributed Bagging, Distributed Stochastic Gradient Descent, Communication Efficient Ensemble Learning, Model Parallelism algorithms are applied to datasets in both Distributed Computing and Traditional Environments.

Keywords: BigData, Distributed Computing, Traditional Computing Environment, Random Forest Classifier, Extra Trees Classifier, Gradient Boosting Regressor, Gradient Boosting Classifier, Voting Classifier, Federated Learning, Distributed Bagging, Distributed Stochastic Gradient Descent, Communication Efficient Ensemble Learning, Model Parallelism.

1. Introduction

The widespread use of social media, cloud computing, sensor networks, content delivery networks, and other new technologies has led to the generation of a significant volume of data daily, which is referred to as "Big Data" [1]. Big data refers to a vast amount of data that is systematically examined to extract valuable insights. Big data is defined by three primary attributes: volume, which refers to the size of the generated data; variety, which encompasses the numerous forms of data such as structured, semi-structured, and unstructured; and velocity, which pertains to the speed at which data is generated and processed. Furthermore, the emergence of big data poses numerous research challenges as conventional computing and database systems cannot handle such large volumes of data. To address the difficulties, Distributed Data Processing Techniques [2] split the task of processing data across multiple less potent computers instead of relying on a single high-powered computer. These algorithms are formulated according to the MapReduce paradigm [3]. MapReduce operates on a cluster of inexpensive computers, making it suitable for Hadoop operations and functions in the processing of massive amounts of data.
The existing machine learning algorithms are applied [24] and compared in both Distributed and Traditional Computing Environments. So, in this paper classifies the structured data with the latest algorithms like Random Forest Classifier, Extra Trees Classifier, Gradient Boosting Regressor, Gradient Boosting Classifier, voting classifier, Federated Learning, Distributed Bagging, Distributed Stochastic Gradient Descent, Communication Efficient Ensemble Learning, Model Parallelism are applied in both Distributed Computing and Traditional Computing Environments to know the best latest algorithm and to know the best environment for the large datasets with less training time.

2. Methodology

In this paper the latest algorithms like Random Forest Classifier, Extra Trees Classifier, Gradient Boosting Regressor, Gradient Boosting Classifier, voting classifier, Federated Learning, Distributed Bagging, Distributed Stochastic Gradient Descent, Communication Efficient Ensemble Learning, Model Parallelism applied on three different structured data to classify the best environment by comparing runtime of environment in both Distributed Computing and Traditional Computing Environments. The step by step workflow is visualised in the figure 1.

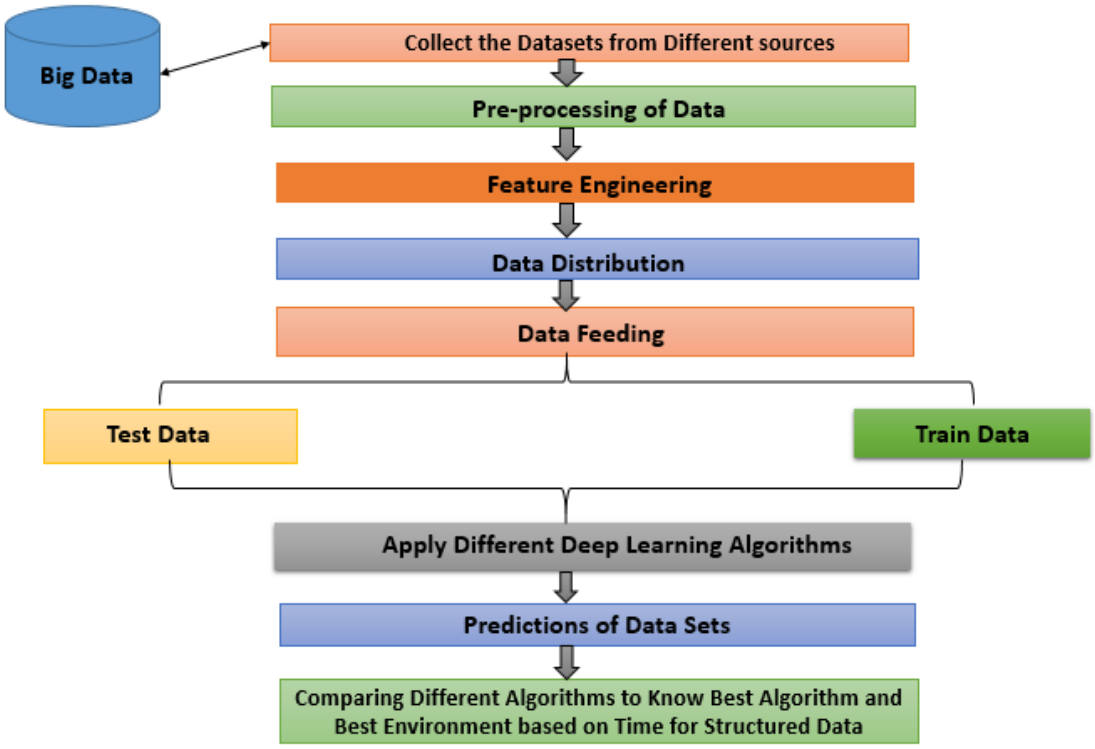


Figure 1: Workflow

2.1. About Datasets

2.1.1. Data Set 1: Login Data Set For Risk Base Authentication

There were almost 33 million login attempts and 3.3 million users on a global scale, and the data collected from this service is one terabyte in size. Information was collected from February 2020 to February 2021. Research and Development for Risk-Based Authentication systems can be accelerated with the use of these data sets. The data was collected from a large-scale single sign-on online system that has more than 3.3 million users from all around the world.

2.1.2. Data Set 2: Internet Traffic Management System

In order to alleviate internet traffic congestion, systems monitor traffic in real-time and implement appropriate measures. Proper maintenance is ensured by capturing the best scores of the Unified Traffic Management system and protocols such as IP, UDP, TCP, HTTP, FTP, DNS, and TFTP.

2.1.3. Data Set 3: Medical Recommendation System

Based on patient feedback, a medical recommendation system suggests providers for a specific ailment. In today's rapidly evolving technological landscape, it is crucial and has the potential to rescue numerous patients' lives. Patients will rate doctors according to how well they accomplish their jobs.

3. Methods

i. Random Forest Classifier

A Random Forest is an ensemble of trees that rely on a set of random factors. In a more formal manner, let's consider a random vector X = (X1,...,Xp)T with p dimensions, which represents the input or predictor variables, and a random variable Y, which represents the response. We

will suppose that there is an unknown joint distribution P_{XY} (X,Y). The objective is to determine a prediction function f(X) that can accurately forecast Y. The prediction function is determined by a loss function, denoted as L(Y,f(X)), and is defined to minimise the expected value of the loss[4].

$$E_{XY}(L(Y, f(X)))$$

The subscripts indicate the expected value based on the joint distribution of X and Y. L(Y,f(X)) can be understood as a quantitative assessment of the proximity between f(X) and Y. It imposes a penalty on values of f(X) that deviate significantly from Y. Common options for loss functions include squared error loss. The loss function for regression is defined as the squared difference between the predicted value (f(X)) and the actual value (Y), denoted as $L(Y,f(X)) = (Y-f(X))^2$. For classification, the loss function is the zero-one loss.

$$L(Y, f(X)) = I(Y \neq f(X)) = \begin{cases} 0 \; if \; Y = f(X \\ 1 \; otherwise. \end{cases}$$

Minimising E_{XY} (L(Y,f(X))) using squared error loss yields the conditional expectation, also referred to as the regression function.

$$f(x) = E(Y|X = x)$$

In a classification scenario, if the range of potential values for Y is represented as Y, the objective of minimising the expected value of the loss function E_{XY} (L(Y,f(X))) for zero-one loss is achieved. $y \epsilon \mathcal{Y}$

$$f(x) = \arg\max P(Y = y \,|X = x)$$

ii. Extra Trees Classifier

The Extra-Trees Algorithm is designed to address the basic batch-mode supervised learning problem, with a specific focus on learning problems that involve a potentially high number of numerical input variables and a single target variable, which can be either categorical or numerical. Next, we proceeded with a methodical empirical assessment using a wide range of classification and regression tasks. We compared this new approach with conventional tree-based methods, evaluating both accuracy and computing efficiency. In the remainder of the paper, the term "attribute" refers to a specific input variable utilised in a supervised learning issue. The candidate characteristics refer to the input variables that are accessible for a specific task. The term "output" is used to refer to the goal variable that describes the problem in supervised learning. When the output variable consists of categories, it is referred to as a classification problem. On the other hand, when the output variable is numerical, it is known as a regression problem [5,6]. The word "learning sample" refers to the observations that are utilised to construct a model, while the term "test sample" refers to the data that are used to calculate the correctness of the model, such as the error rate or mean square error. The variable N represents the size of the learning sample, namely the number of observations. On the other hand, the variable n represents the number of candidate characteristics, which corresponds to the dimensionality of the input space.

iii. Gradient Boosting

The gradient boost approach aids in reducing the bias error of the model. The fundamental concept underlying this technique is to construct models in a sequential manner, with each succeeding model aiming to minimise the errors made by the prior model. However, what is the method by which we accomplish that? What strategies can be employed to minimise the occurrence of errors? This is accomplished by constructing a novel model based on the errors or residuals of the preceding model [7].

For continuous target columns, Gradient Boosting Regressor is employed, whereas for classification problems, Gradient Boosting Classifier is utilised. The sole distinction between the two lies in the "Loss function". The goal is to reduce this loss function by incorporating weak learners through the use of gradient descent. Regression problems utilise various loss functions such as Mean Squared Error (MSE), which is based on the loss function. On the other hand, classification issues employ distinct functions like log-likelihood.

1.Gradient Boosting Regressor

The Gradient Boosting Regressor (GBR) is an ensemble model that consists of a series of tree models placed in a sequential manner. Each subsequent model learns from the errors made by the previous model through an iterative process. This machine learning model use the technique of "boosting" to combine weak prediction models, typically decision trees, in order to create a more resilient and effective model [8]. A GBR with M trees can be defined as follows: hm represents a weak learner that has low individual performance, whereas γm is a scaling factor that determines the contribution of each tree to the overall model. GBR employs the gradient descent loss function to minimise errors by iteratively updating the starting estimation with the new estimation. Therefore, a conclusive model is constructed by combining all initial estimations with appropriate weights. The GBR model utilised in this study is based on the Gradient Boosting Regressor technique proposed by [9].

$$f_M(x_j) = \sum_{m}^{M} \gamma_m h_m(x_j)$$

The model's error in regression analysis refers to the discrepancy between the observed data points and the line of best fit generated by the algorithm. The array contains the elements. The inaccuracy of the model will be determined based on the following criteria.

1. **Mean Absolute Error (MAE):** is a statistical measure that calculates the average of the absolute values of the mistakes, which reflect the extent of divergence from the real probability. The mathematical expression for Mean Absolute Error (MAE) is given by the formula:

$$MAE = \frac{\sum_{i=1}^{n} |y_i - x_i|}{n}$$

2. **Root Mean Squared Error (RMSE):** is a widely used metric for evaluating the performance of models. It is valuable because it can be understood as the standard deviation of the prediction errors. The expression is represented as follows:

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

3. **Execution Time:** refers to the duration it takes for the model to learn from the input logs and provide predictions.

2. Gradient Boosting Classifier

Both the loss function and the base-learner models can be specified at will. Obtaining the parameter estimates can be challenging in practice, especially when dealing with a customised loss function Ψ(y, f) and/or a customised base-learner h(x, θ). In order to address this issue, a suggestion was made to select a new function h(x, θt) that is most closely aligned with the negative gradient $\{g_t(x_i)\}^N{}_{i=1}$ .The value of Ni is equal to 1 for the observed data[10].

$$\text{gt(x)} = E_y \left[\frac{\partial \Psi(y, f(x))}{\partial f(x)} \mid x \right]_{f(x) = \hat{f}^{t-1}(x)}$$

Instead of seeking the overall solution for the increase in boost in the function space, one can opt for the new function increment that is most closely related to $g_t(x)$. This allows for the substitution of a potentially challenging optimisation problem with the traditional least-squares minimization problem:

$$\left(\rho_{t,\ \theta_t}\right) = \arg min_{\rho,\theta} \sum_{i=1}^{N} [-g_t(x_i) + ph(x_i, \theta)]^2$$

iv. Voting Classifier

This approach is the most straightforward strategy in which each voter is entitled to select only one option, specifically their most desired one. The victor is the contender who possesses the highest aggregate number of votes. The given text is the list [11, 12]. In this scenario, if 45% of voters select alternative A, 25% choose B, and 30% choose C, then alternative A is the victor. Condorcet's July theorem states that if there are m Voters who make decisions through simple majority voting, and each has a probability p of making the correct decision, then the probability of the entire jury making the correct decision is:

$$p_m = \sum_{i=\lceil m/2 \rceil}^{m} \left(\frac{m!}{(m-i)! \cdot i!} \right) \cdot p^i \cdot (1-p)^{m-i}$$

Therefore, if the value of p is greater than 0.5, then the value of $p_m > p$ will be greater than p. This implies that the collective has a greater likelihood of making the accurate determination compared to any individual voter. As the number of voters' increases, the likelihood of the ensemble making the correct decision also increases. When the value of m approaches infinity, the probability $m \to \infty, p_m \to 1$.

v. Federated Learning

Federated Learning (FL) is a secure and decentralised learning framework that allows the creation of a virtual model to handle the issue of scattered clients working together without the need to share sensitive raw data [13]. The virtual model is an efficient global model that combines data from all participants, allowing each person to achieve their individual goals utilising the model. FL can ensure that the results of this modelling closely resemble the usual centralised training paradigm [14], where data from numerous clients are consolidated in a central server for modelling [15]. Within a federated system, it is commonly presumed that members possess identical identities and endorse the implementation of mutually agreed-upon data policies. As the data is not transmitted directly, it does not impact data standards or endanger user privacy.

To begin, we establish the fundamental notion of FL: Consider a scenario where there are N participants in Federated Learning (FL), denoted as $\{P1, P2, \cdots, PN\}$, who all like to combine their data in order to build a unified model. An often employed method involves aggregating all the data into a single set, denoted as total data $P=P1 \cup P2 \cup \cdots \cup PN$, to train a model $Msum$ with a performance of $Vsum$. Federated Learning (FL) is a framework in which participants collaboratively train a shared model $Mfed$ while maintaining the privacy of their individual data. The performance of the model is evaluated using $Vfed$. Assuming ε is a non-negative value, the performance loss of the FL model can be represented by

$$|V_{fed} - V_{sum}| < \epsilon$$

The process of learning in federated learning is accomplished by minimising a loss function, which is computed on each client using a weighted aggregation technique. The objective of FL is to minimise the following function:

$$\min f(\omega) = \sum_{k=1}^{n} \frac{n_k}{n} F_k(\omega)$$

Where N is the number of clients, nk is the amount of data on the k^{th} client, and $Fk(w)$ is the local objective function of the k^{th} client.

vi. Distributed Stochastic Gradient Descent

Gradient descent is a highly favoured approach for optimisation and is widely used for optimising neural networks. Simultaneously, many modern Deep Learning libraries provide implementations of different methods to optimise gradient descent [16]. These algorithms are frequently employed as black-box optimizers, as it is difficult to find practical explanations of their advantages and disadvantages.

Let K be a positive integer representing discrete time steps. The variable x_n(k) represents agent n's estimate of the minimizer of equation at iteration k. The D-SGD algorithm is defined on a per-agent basis using recursion.

$$x_n(k+1) = x_n(k) - \propto_k \left(\nabla f_n\big(x_n(k)\big) + \xi_n(k+1)\right) + \beta_k \sum_{\ell \in \Omega_n} (x_\ell(k) - x_n(k))$$

For n =1,.....N, where $\{\propto_k\}_{k \geq 1}, \{\beta_k\}_{k \geq 1} \subset (0, 1]$ are scalar weight parameters, $\xi_n(k)$ represents zero-mean noise, and Ω_n denotes the set of neighbors of agent n in the graph G. The method is initialized by setting the vector $((x_n(0))_{n=1}^{N})$ to a specified point $x_0 \in \mathbb{R}^{N\,d}$.

vii. Communication Efficient Ensemble Learning

Let's consider a set H consisting of q pre-trained hypotheses H = (h1, ..., hq) which can be either predictors or estimators depending on the job. It is preferable to have a reliable predictor for each domain k. However, these predictors can be trained using many methods, such as user data or public data [17]. The objective is to acquire a collection of weights α = {α1, ..., αq} in order to create an ensemble $\sum_{k=1}^{q} \propto_k h_k$ $\sum_k \alpha_k = 1$. that reduces the standard or agnostic loss to a minimum. In addition, we place primary emphasis on density estimate, assuming that the set of hypotheses, denoted as H, is defined as follows:

$H = \left\{\sum_{k=1}^{q} \alpha_k h_k : \alpha_k \geq 0,\ \forall k,\ \sum_{k=1}^{q} \propto_k = 1\right\}$.

By considering this particular collection of assumptions will minimise both the standard loss and the agnostic loss.

viii. Model Parallelism

Model parallelism in neural networks involves dividing the model into parts and distributing those parts across many computing resources. This technique has the potential to provide advantages in terms of both model throughput and reducing memory needs per device [18]. In order to provide a clearer understanding of model parallelism, we will begin by presenting a computational framework for analysing neural networks. This framework will serve as a foundation for discussing model parallelism, which involves neural networks functioning in the Single Instruction Multiple Data (SIMD) format. Subsequently, we address the initial study

inquiry, namely, "What are the various forms of model parallelism?" taking into account both theoretical and implementation aspects. This algorithm differentiates between two distinct phases: training and inference. During the training phase, we develop a model using a specific dataset known as the learning set [19]. During inference, we assign the trained model the responsibility of making predictions on new, previously unknown data. A neural network is one example of such models, and they are highly effective at handling difficult prediction tasks, surpassing other existing methods. The input tensor **X** and the output tensor **Y**. Furthermore, we make a clear distinction between parameter tensors and activation tensors. Parameter tensors are fixed inputs to operators, whereas activation tensors are the output of such operators. Operators serve as the functional components of the neural network. The user's text is [20].

In the operator graph $\mathcal{O}=(V, E)$ of a given neural network, each node $vi \in V$ can be classified as either an operator oi, which is accompanied by an activation tensor $\mathbf{T}oi$, or a tensor $\mathbf{T}i$. Each edge represents the tensor linked to vi and serves as an input for the operator node oj.

Based on this representation, we establish two workloads: the forward pass and the backward pass. The forward pass involves taking the input tensor **X** and calculating all the activations, which leads to the output tensor **Y**. The backward pass modifies the parameter tensors by utilizing the back-propagation process.

- The backward pass begins at the output **Y** and progresses towards **X**, establishing its dependence on the outcome of the forward pass.
- It necessitates the activation tensor of each operator to determine the appropriate parameter updates.

The training process involves b iterations of forward passes followed by b iterations of backward passes, where b is the batch size. Inference solely involves doing forward passes [21-23].

3.Experimental Analysis

In this work experimental analysis is done in two environments like distributed environment and traditional computing environment. For distributed environment Spark runtime environment is used whereas for traditional computing environment python programing environment is used. Here the analysis is done for three different types of structured datasets: Login Data Set for Risk Base Authentication, Internet Traffic Management System, Medical Recommendations System. The datasets have terabytes of data so; in distributed environment the data is divided into chunks of volumes for easy access. Whereas in traditional computing environment the Google Colab Python platform is used for easy access of data.

At initial step the pre-processing is done for three datasets individually with procedure.

- First, data cleaning with multiple sub categories like: renaming the columns, to find and remove the null columns in dataset, then checking the duplicate records, later changing the data frame from strings to numeric for easy training and testing dataset.
- Next feature Engineering is performed to find the input features because to know which data is suitable for each column and is passed to the different deep learning algorithms.
- After pre-processing, the three datasets are individually spited into 70% training and 30% testing to overcome the overfitting problems in both environments.
- Then latest deep learning algorithms like Random Forest Classifier, Gradient Boost Regressor, Gradient Boost Classifier, Voting Classifier, Extra Tree Classifier, Federated Learning, Distributed Bagging, Distributed Stochastic Gradient Descent, Communication Efficient Ensemble Learning, Model Parallelism on this dataset in both the environments to

know the best environment for processing of large structured datasets and the best proposed algorithm.

The results of three datasets are:

3.1. Data Set: LOGIN DATA SET FOR RISK BASE AUTHENTICATION

After applying different latest Deep Learning Algorithms on login data set for risk base authentication in both Distributed Computing Environment and Traditional Computing Environment, the Accuracy, Time to Train, predict in both Distributed Environment and Traditional Computing Environment as shown in table 1&2.

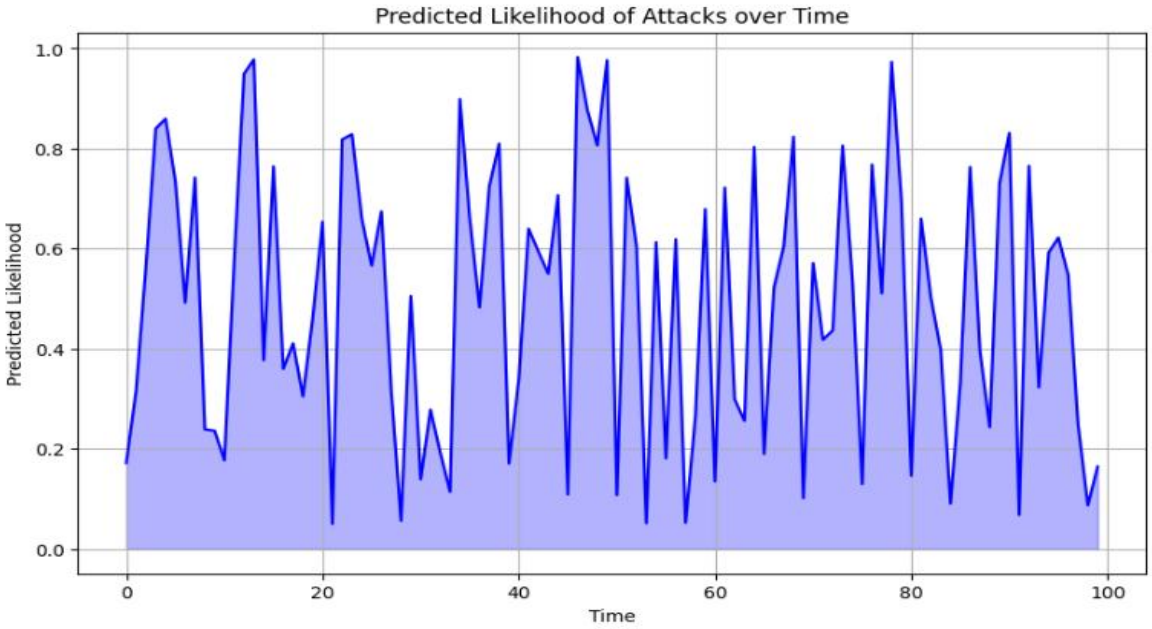
Table 1: Comparison of Algorithms in Distributed Computing Environment

Algorithms	Accuracy	Recall	Precision	F1-Score	Time to train in sec	Time to prediction in sec	Total time in sec
Random Forest Classifier	93.06%	93.06%	93.86%	93.06%	13.0	16.0	29.0
Gradient Boost Regressor	95.06%	95.06%	95.86%	95.06%	6.0	14.0	20.0
Gradient Boost Classifier	89.51%	89.51%	89.59%	89.61%	9.0	14.0	23.0
Voting classifier	66.26%	66.15%	66.95%	61.37%	7.4	12.4	19.8
Extra Tree Classifier	81.51%	81.95%	81.96%	81.95%	46.0	9.0	55.0
Federated Learning	89.63%	89.65%	89.56%	87.26%	5.9	8.4	14.2
Distributed Bagging	98.96%	98.97%	98.96%	97.87%	8.8	10.6	19.4
Distributed Stochastic Gradient Descent	91.34%	91.69%	91.32%	88.65%	12.0	6.2	18.3
Communication-Efficient Ensemble Learning	97.78%	97.78%	97.96%	95.27%	15.0	14.0	29.0
Model Parallelism	98.95%	98.59%	98.96%	98.06%	7.3	4.3	11.6

By observing table 1, in Distributed Environment Model Parallelism is the best algorithm with the Accuracy of 98.95% and Time to Train 7.3 Sec, Time to Predict 4.3 Sec.

Table 2: Comparison of Algorithms in Traditional Computing Environment

By observing Table 2, in Traditional Computing Model Parallelism is the best algorithm with the Accuracy of 96.91% and Time to Train 10.3 Sec, Time to Predict 6.1Sec.



So, by comparing both Distributed Computing and Traditional Computing Environments, the Distributed Environment is the best because the total runtime of training the dataset is less 11.6 Sec while compared to the traditional computing Environment 16.4.
The prediction of dataset analysed in both environments is:
At first predicted likelihood of attacks for each timestamp is visualised in figure 6.

Figure 6: Predicted Likelihood of Attacks for Each Timestamp
In figure 6, X-axis is Time and Y-axis is Prediction likelihood by observing the figure across the globe at every 15seconds the internet attacks are happening.
Next predicted the distribution of attacks by OS name and OS version is visualised in figure 7.

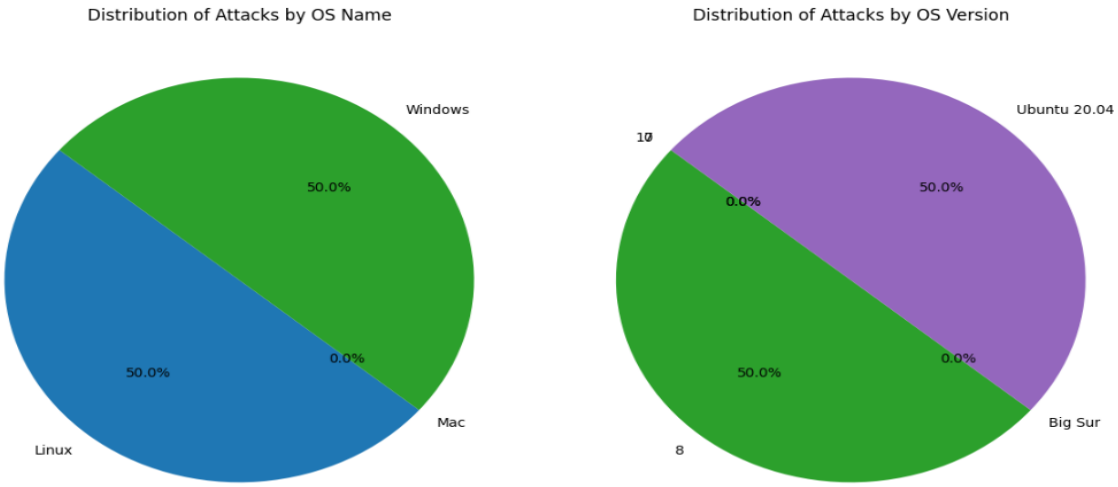


Figure 7: Distribution of Attacks by OS Name and OS Version

In figure 7, X-axis is name of the software, version of the software and Y-axis is attacks occurring at timestamp the two separate pie charts are shown the distribution of attacks by OS

name and OS version. In distributed OS name Windows has high rate of attacks and in OS version windows 8 has the highest attacks.

a.DATA SET: INTERNET TRAFFIC MANAGEMENT SYSTEM

After applying different latest Deep Learning Algorithms on the Internet Traffic Management System in both Distributed Computing Environment and Traditional Computing Environment. Observed the Accuracy, Time to Train, predict in both Distributed Environment and Traditional Computing Environment as shown in Table 3&4.

Table 3: Comparison of Algorithms in Distributed Computing Environment

Algorithms	Accuracy	Recall	Precision	F1-Score	Time to train in sec	Time to predict in sec	Total time in sec
RandomForest Classifier	97.95%	97.70 %	97.70%	96.60 %	13.1	23.9	37.0
Extra Tree Classifier	95.80 %	95.77 %	95.77%	94.66 %	10.4	23.8	34.2
GradientBoost Classifier	81.90%	81.90 %	81.89%	79.89 %	8.7	22.4	31.1
Voting classifier	98.90%	98.90 %	98.90%	96.89 %	5.7	38.4	44.1

Algorithms	Accuracy			Recall	Precision	F1-Score	Time to train in sec	Time to predict in sec	Total time in sec
GradientBoost Regressor	92.12 %	93.35 %	94.56 %	89.45 %	16.6	19.2		35.8	
Federated Learning	86.18 %	86.12 %	86.25 %	84.98 %	15.0	14.2		29.2	
Distributed Bagging	91.78 %	91.76 %	91.39 %	89.56 %	8.9	9.4		18.2	
Distributed Stohastic Gradient Descent	85.41 %	85.41 %	85.41 %	83.65 %	16.0	6.1		22.1	
Communication-Efficient Ensemble Learning	69.14 %	69.78 %	69.52 %	66.89 %	8.0	11.4		19.3	
Model Parallelism	96.91 %	96.98 %	96.95 %	92.67 %	10.3	6.1		16.4	

Algorithms	Accuracy	Recall	Precision	F1-Score	Time to train in sec	Time to predict in sec	Total time in sec
Random Forest Classifier	95.06%	95.06 %	95.86%	95.06 %	6.0	14.0	20.0

Gradient Boost Regressor			88.06%	88.06 %	88.56%	88.06 %	14.0	12.0	26.0
Gradient Boost Classifier			86.95%	86.95 %	86.96%	86.95 %	15.0	11.0	26.0
Voting classifier			97.51%	97.51 %	97.59%	97.51 %	10.0	14.0	14.0
Extra Tree Classifier			79.51%	79.51 %	79.59%	79.51 %	46.0	19.0	65.0
Federated Learning			82.32%	82.56 %	82.45%	80.56 %	2.7	2.3	5.0
Distributed Bagging			94.15%	94.16 %	94.55%	96.32 %	4.9	1.6	6.4
Distributed Stohastic Gradient Descent	98.47 %	89.84 %	82.45%	80.56 %	2.0	3.3		5.4	
Communication-Efficient Ensemble Learning	88.78 %	88.84 %	88.45%	86.56 %	1.7	1.4		3.1	
Model Parallelism	98.45 %	98.12 %	98.44%	95.23 %	1.6	1.1		2.7	

By observing table 1, in Distributed Environment Model Parallelism is the best algorithm with an Accuracy of 98.45% and Time to Train 1.6 Sec, Time to Predict 1.1 Sec.

Table 4: Comparison of Algorithms in Traditional Computing Environment

By observing table 2, In Traditional Computing Model Parallelism is the best algorithm with the Accuracy of 93.13% and Time to Train 5.7 Sec, Time to Predict 4.1 Sec.
So, by comparing both Distributed Environment and Traditional Computing Environments, the Distributed Environment is the best because the total runtime of training the dataset is less 2.7 sec while compared to the Traditional Computing Environment is 9.8 sec.
The prediction of dataset analysed in both environments is:
At first predicated the of average drops happening in service within an Internet Traffic Management System is visualised in figure 8.

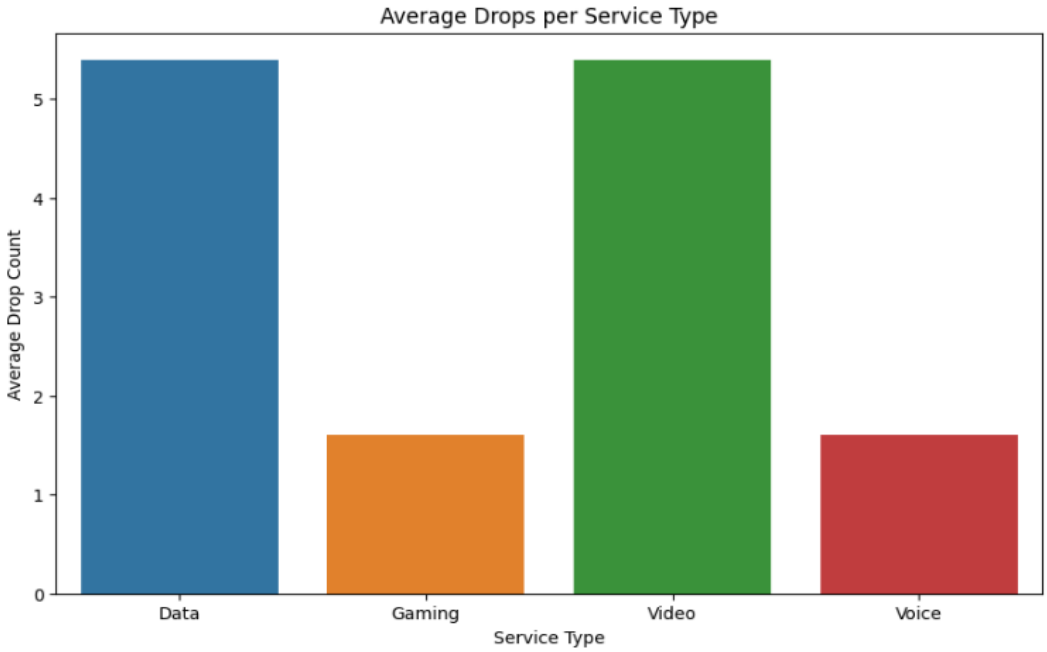


Figure 8: Average Drops Happening in Service Within an Internet Traffic Management System

In figure 8, X-axis is Service Time and Y-axis is Average Drop count by observing while data browsing and playing games internet drops is high.

Next predicted the distribution and density of the data, providing insights into the variability of user logins for each service is visualised in figure 9.

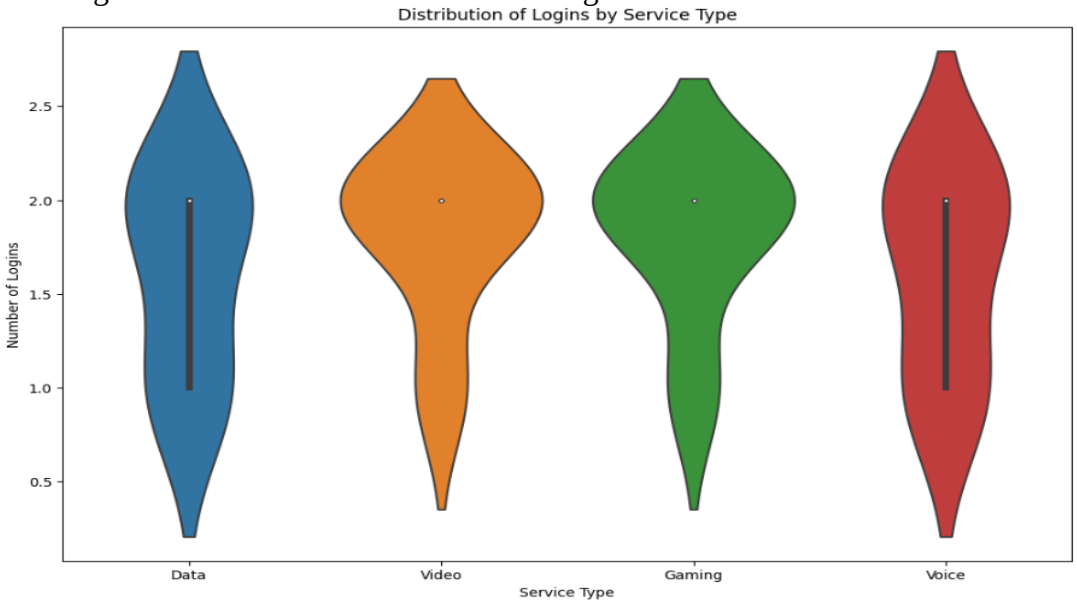


Figure 9: The Distribution and Density of the Data, Providing Insights into the Variability of User Logins for Each Service

In figure 9, X-axis is service type and Y-axis is number of logins visualizes that while data browsing and voice connection calls the distribution and density of user logins are high for each service.

b. DATA SET: MEDICAL RECOMMENDATIONS SYSTEM

After applying different latest Deep Learning Algorithms on the Medical Recommendations systems in both Distributed Computing Environment and Traditional Computing Environment. Observed the Accuracy, Time to Train, predict in both Distributed Environment and Traditional Computing Environment as shown in Table 5&6.

Table 5: Comparison of Algorithms in Distributed Computing Environment

Algorithms	Accuracy	Recall	Precision	F1-Score	Time to train in sec	Time to prediction in sec	Total time in sec
Random Forest Classifier	75.36%	75.60%	75.69%	73.66%	4.4	19.9	24.3
Gradient Boost Regressor	65.36%	65.70%	66.76%	64.74%	5.4	13.9	19.3
Gradient Boost Classifier	97.96%	97.69%	97.69%	96.60%	11.4	21.9	33.2
Voting classifier	71.80%	71.77%	71.77%	69.60%	9.4	32.9	42.3
Extra Tree Classifier	86.97%	86.77%	86.77%	76.60%	10.4	20.9	31.3
Federated Learning	93.14%	93.14%	93.25%	90.15%	3.7	4.7	8.4
Distributed Bagging	97.15%	97.14%	97.20%	95.12%	3.1	1.3	4.4
Distributed Stohastic Gradient Descent	90.65%	90.65%	90.23%	88.54%	2.6	0.6	3.1
Communication-Efficient Ensemble Learning	75.12%	74.57%	74.89%	70.19%	2.8	2.3	5.1
Model Parallelism	98.14%	98.12%	98.44%	96.23%	1.9	0.5	2.4

By observing table 5, in Distributed Environment Model Parallelism is the best algorithm with the Accuracy of 98.14% and Time to Train 1.9 Sec, Time to Predict 0.5 Sec

Table 6: Comparison of Algorithms in Traditional Computing Environment

By observing 6, in Traditional Computing Model Parallelism is the best algorithm with the Accuracy of 91.76% and Time to Train 10.1 Sec, Time to Predict 12.2 Sec.

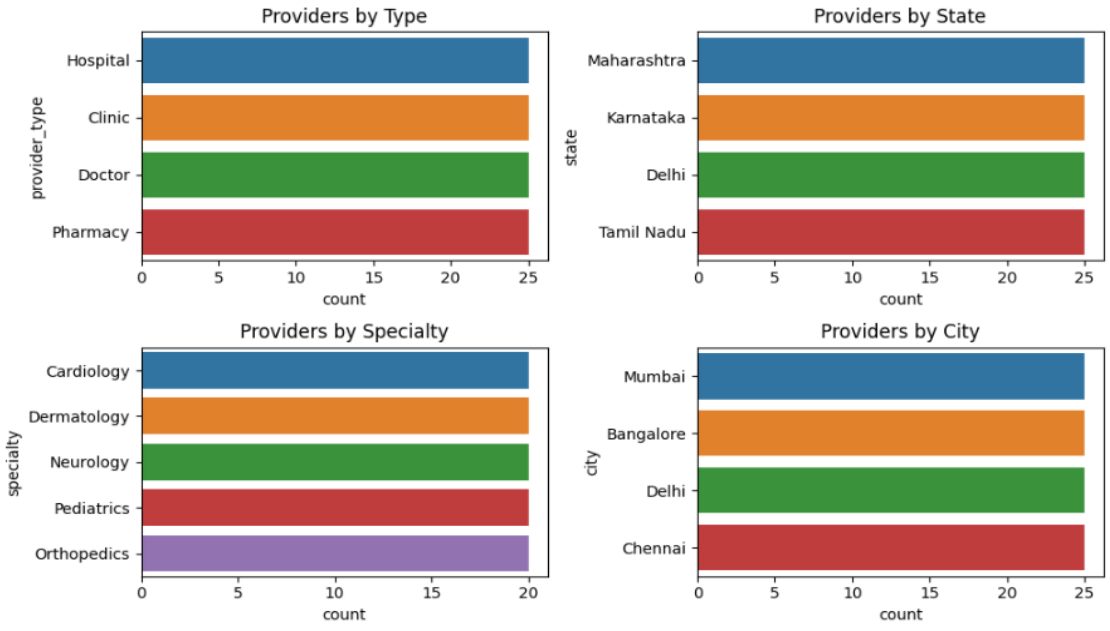
So by comparing both Distributed Environment and Traditional Computing Environments, the Distributed Environment is the best because the total runtime of training the dataset is less 2.4 sec while compared to the Traditional Computing Environment is 22.2 sec.

The prediction of dataset analysed in both environments is:

At first the medical service providers in India is predicated is visualised in figure 10.

Figure 10: Medical Service Providers in India
In figure 10, X-axis is count and Y-axis is provider type, state, city, Speciality the hospitals,

Algorithms	Accuracy	Recall	Precision	F1-Score	Time to train in sec	Time to predict in sec	Total time in sec
Random Forest Classifier	91.03%	65.06%	75.16%	86.25%	8.0	24.0	32.0
Extra Tree Classifier	82.16%	77.01%	78.26%	70.02%	20.0	33.0	53.0
Gradient Boost Classifier	66.29%	61.95%	65.16%	63.15%	14.0	21.0	35.0
Voting classifier	93.13%	92.11%	89.29%	88.51%	20.4	24.0	44.4
Gradient Boost Regressor	73.23 %	70. 12%	69.59%	72.41%	29.0	40.0	69.0
Federated Learning	74.68%	74.23%	74.12%	71.39%	5.7	4.1	9.8
Distributed Bagging	86.87%	86.77%	86.77%	76.60%	10.4	20.9	31.3
Distributed Stohastic Gradient Descent	84.52%	89.85%	89.33%	87.46%	10.6	10.9	21.5
Communication-Efficient Ensemble Learning	75.36%	75.76%	75.79%	73.66%	4.4	14.9	19.3
Model Parallelism	91.76%	91.86%	91.33%	90.23%	10.1	12.2	22.2



clinic, doctor, pharmacy's with speciality are considered by observing medical services are well established in the main states and city's so there is need of improving the medical services in the urban areas.
Next analysed area-wise clinics in India is visualised in figure 11.

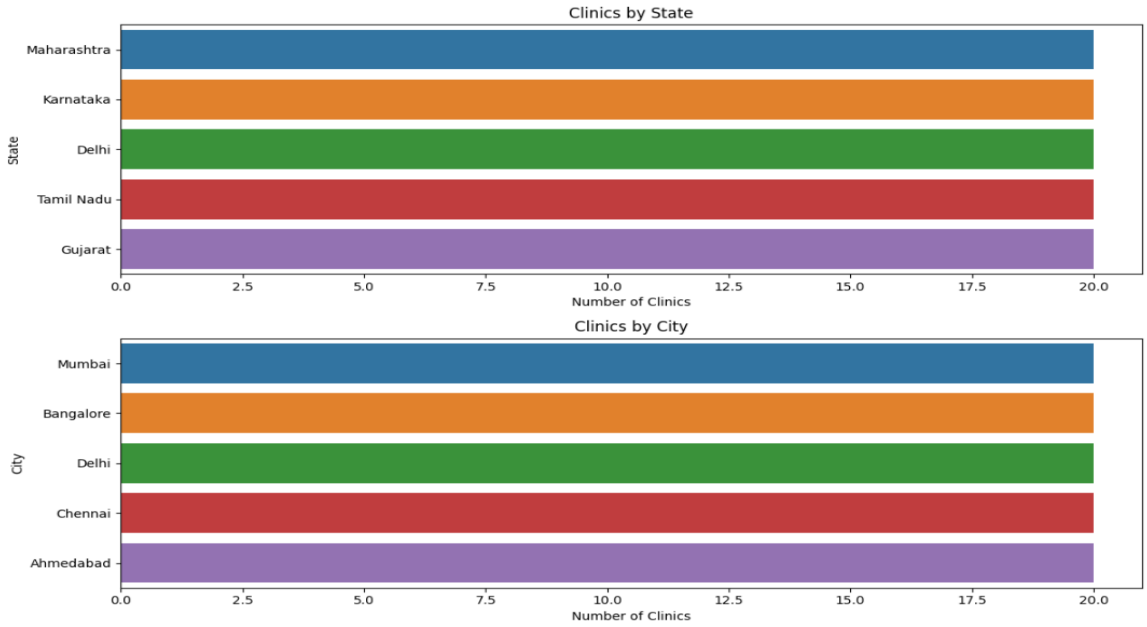


Figure 11: Area of Wise Clinics in India
In figure 11, X axis is number of clinics and Y-axis is state, City in every state observed that the main cities are well equipped with clinics and in urban areas medical services have to be improved a lot.
So by comparing both Distributed Computing and Traditional Computing Environments analysis of different structured datasets by applying different latest deep learning algorithms like: Random Forest Classifier, Gradient Boost Regressor, Gradient Boost Classifier, Voting Classifier, Extra Tree Classifier, Federated Learning, Distributed Bagging, Distributed Stochastic Gradient Descent, Communication Efficient Ensemble Learning, Model Parallelism. By comparing all the algorithms in both Distributed Computing and Traditional Computing Environments Model Parallelism algorithms is the best algorithm in terms of accuracy. In terms of training time, Distributed Computing Environment is the best environment for large datasets.

4. Conclusion

The utilization of big data analysis has become crucial in contemporary company operations, enabling organizations to acquire valuable insights and make well-informed decisions based on data. To effectively manage the large quantity, varied categories, and rapid pace of big data. On the other hand, distributed computing and deep learning algorithms have the promise to effectively tackle the challenges of analyzing and learning from large amounts of input data. More precisely, it assists in the automated extraction of intricate data representations from extensive quantities. So, here three structured datasets are taken and to these latest algorithms like Random Forest Classifier, Extra Trees Classifier, Gradient Boosting Regressor, Gradient Boosting Classifier, voting classifier, Federated Learning, Distributed Bagging, Distributed Stochastic Gradient Descent, Communication, Efficient Ensemble Learning, Model Parallelism algorithms are applied in both Distributed Computing and Traditional Computing

Environments. Then compared the latest deep learning algorithms, Model Parallelism is the best algorithm in both environments with high accuracy. In terms of training time, distributed computing environment is the best and efficient environment for running large data.

5. References

1. Wu, C.; Birch, D.; Silva, D.; Lee, C.-H.; Tsinalis, O.; Guo, Y.: Concinnity: (2014) "A generic platform for big sensor data applications", IEEE in Cloud Comput. **1**(2), 42–50.
2. Chen, T.Y.; Wei, H.W.; Wei, M.F.; Chen, Y.J.; Hsu, T.s.; Shih, W.K.: LaSA: (2013) "A locality-aware scheduling algorithm for Hadoop-MapReduce resource assignment", In International Conference on Collaboration Technologies and Systems (CTS), San.
3. Diego, CA, pp. 342–346 (2015) 11. Krishnan, N.; Baron, D.: A universal parallel two-pass MDL context tree compression algorithm. IEEE J. Sel. Top. Signal Process. **9**(4), 741–748
4. Zhang, H.; Chen, G.; Ooi, B.C.; Tan, K.L.; Zhang, M.: (2015) "In-memory big data management and processing: a survey", IEEETrans. Knowl. Data Eng. **27**(7), 1920–1948.
5. Abellán, J., Mantas, C.J., (2014), "Improving experimental studies about ensembles of classifiers for bankruptcy prediction and credit scoring", Expert Syst. Appl. 41 (8), 3825–3830.
6. Alshazly, H., Linse, C., Barth, E., Martinetz, T., (2019), "Ensembles of deep learning models and transfer learning for ear recognition", Sensors 19 (19), 4139.
7. Bharathidason, S., Venkataeswaran, C.J., (2014), "Improving classification accuracy based on random forest model with uncorrelated high performing trees", Int. J. Comput. Appl 101 (13), 26–30.
8. Delgado, R., (2022), "A semi-hard voting combiner scheme to ensemble multi-class probabilistic classifiers", Appl. Intell. 52 (4), 3653–3677.
9. Anifowose, F., Labadin, J., Abdulraheem, A., (2013), "Ensemble model of artificial neural networks with randomized number of hidden neurons", In 8th International Conference on Information Technology in Asia (CITA). IEEE, pp. 1– 5.
10. Freund, Y., Schapire, R.E., et al. (1996), "Experiments with a new boosting algorithm", pp. 148–156.
11. Haralabopoulos, G., Anagnostopoulos, I., McAuley, D., (2020), "Ensemble deep learning for multi-label binary classification of user-generated content", Algorithms 13 (4), 83.
12. Hormozi, E., Akbari, M.K., Hormozi, H., Javan, M.S., (2013), "Accuracy evaluation of a credit card fraud detection system on hadoop mapreduce", The 5th Conference on Information and Knowledge Technology. IEEE, pp. 35–39.
13. Kanakaraj, M., Guddeti, R.M.R., (2015), "Performance analysis of ensemble methods on twitter sentiment analysis using NLP techniques", In: Proceedings of the 2015 IEEE 9th International Conference on Semantic Computing (IEEE ICSC 2015). IEEE, pp. 169–170.
14. Kang, S., Kang, P., KO, T., Cho, S., Rhee, S.-J., Yu, K.-S., (2015), "An efficient and effective ensemble of support vector machines for anti-diabetic drug failure prediction." Expert Syst. Appl. 42 (9), 4265–4273.
15. Dai, Q., (2013), "A competitive ensemble pruning approach based on cross-validation technique", Knowl.-Based Syst. 37, 394–414.

16. Delgado, R., (2022), "A semi-hard voting combiner scheme to ensemble multi-class probabilistic classifiers", Appl. Intell. 52 (4), 3653–3677.
17. Latif-Shabgahi, G.-R., (2004), A novel algorithm for weighted average voting used in fault tolerant computing systems. Microprocess. Microsyst. 28 (7), 357–361.
18. Livieris, I.E., Iliadis, L., Pintelas, P., (2020), "On ensemble techniques of weight- constrained neural networks", Evolv. Syst., 1–13.
19. Onan, A., Korukog˘lu, S., Bulut, H., (2016), "A multi objective weighted voting ensemble classifier based on differential evolution algorithm for text sentiment classification", Expert Syst. Appl. 62, 1–16.
20. Rokach, L., (2019), Ensemble learning: Pattern classification using ensemble methods. World Sci. 85.
21. Xia, R., Zong, C., Li, S., (2011), "Ensemble of feature sets and classification algorithms for sentiment classification", Informat. Sci. 181 (6), 1138–1152.
22. Zhang, J., Zhang, W., Song, R., Ma, L., Li, Y., (2020). "Grasp for stacking via deep reinforcement learning", In IEEE International Conference on Robotics and Automation (ICRA). IEEE, pp. 2543–2549.
23. Zhang, W., Zou, H., Luo, L., Liu, Q., Wu, W., Xiao, W., (2016), Predicting potential side effects of drugs by recommender methods and ensemble learning. Neurocomputing 173, 979–987.
24. M. Bhargavi Krishna, Prof. S. Jyothi (2023) "Performance of Machine Learning Algorithms in Distributed Environment", ISSN: 2094-0343 2326-9865 Published by A Study Mathematical Statistician and Engineering Applications, vol.72 No.1.