electronics

MDPI

Article

Quality of Service Generalization using Parallel Turing Integration Paradigm to Support Machine Learning

Abdul Razaque [1], Mohamed Ben Haj Frej [2,*], Gulnara Bektemyssova [3,*], Muder Almi'ani [4,*], Fathi Amsaad [5], Aziz Alotaibi [6], Noor Z. Jhanjhi [7], Mohsin Ali [3], Saule Amanzholova [1] and Majid Alshammari [6]

1 Department of Cyber Security, International Information Technology University, Almaty 050000, Kazakhstan
2 Department of Computer Science and Engineering, University of Bridgeport, Bridgeport, CT 06604, USA
3 Department of Computer Engineering, International Information Technology University, Almaty 050000, Kazakhstan
4 Department of Management Information System (MIS), Gulf University for Science and Technology, Kuwait City 32093, Kuwait
5 Department of Computer Science, Joshi Research Center, University of Wright, Dayton, OH 45435, USA
6 Computers and Information Technology College, Taif University, Taif 21974, Saudi Arabia
7 School of Computer Science, Taylor's University, Subang Jaya 47500, Malaysia
* Correspondence: mhenhaj@bridgeport.edu (M.B.H.F.); g.bektemisova@iitu.edu.kz (G.B.); almiani.m@gust.edu.kw (M.A.)

Abstract: The Quality-of-Service (QoS) provision in machine learning is affected by lesser accuracy, noise, random error, and weak generalization (ML). The Parallel Turing Integration Paradigm (PTIP) is introduced as a solution to lower accuracy and weak generalization. A logical table (LT) is part of the PTIP and is used to store datasets. The PTIP has elements that enhance classifier learning, enhance 3-D cube logic for security provision, and balance the engineering process of paradigms. The probability weightage function for adding and removing algorithms during the training phase is included in the PTIP. Additionally, it uses local and global error functions to limit overconfidence and underconfidence in learning processes. By utilizing the local gain (LG) and global gain (GG), the optimization of the model's constituent parts is validated. By blending the sub-algorithms with a new dataset in a foretelling and realistic setting, the PTIP validation is further ensured. A mathematical modeling technique is used to ascertain the efficacy of the proposed PTIP. The results of the testing show that the proposed PTIP obtains lower relative accuracy of 38.76% with error bounds reflection. The lower relative accuracy with low GG is considered good. The PTIP also obtains 70.5% relative accuracy with high GG, which is considered an acceptable accuracy. Moreover, the PTIP gets better accuracy of 99.91% with a 100% fitness factor. Finally, the proposed PTIP is compared with cutting-edge, well-established models and algorithms based on different state-of-the-art parameters (e.g., relative accuracy, accuracy with fitness factor, fitness process, error reduction, and generalization measurement). The results confirm that the proposed PTIP demonstrates better results as compared to contending models and algorithms.

Keywords: predictive modeling; blending algorithm; data mining; optimum fitting; tuning integration; overfitting; logical table; QoS

Citation: Razaque, A.; Frej, M.B.H.; Bektemyssova, G.; Almi'ani, M.; Amsaad, F.; Alotaibi, A.; Jhanjhi, N.Z.; Ali, M.; Amanzholova, S.; Alshammari, M. Quality of Service Generalization using Parallel Turing Integration Paradigm to Support Machine Learning. *Electronics* **2023**, *12*, 1129. https://doi.org/10.3390/electronics12051129

Academic Editor: Andrei Kelarev

Received: 13 January 2023
Revised: 8 February 2023
Accepted: 22 February 2023
Published: 25 February 2023

1. Introduction

The performance of machine learning depends significantly on the properties of the algorithms [1]. ML is made up of various cutting-edge algorithms that correctly predict events and help systems make informed judgments [2,3]. However, some of the current issues in ML include under-fitting, overfitting, bias, inconsistencies, and reduced accuracy [4]. Numerous algorithms have been developed, although they may not be entirely capable of solving these problems [5–7]. However, given the advent of digital and e-data,

particularly in the fields of Big data and social networking data, ML has made appreciable development in data science [8,9].

Predictive analytics in ML uses a variety of models, approaches, and algorithms [10,11]. Recent research has focused on blending models and features [12–14]. To tackle bias, mistakes, overfitting/underfitting, and poor generalization, this work presents a novel technique of blending, in contrast to other recent advancements in data science research.

Many computations are performed concurrently to provide the solution more quickly. Therefore, faster computation and QoS provisioning can be made feasible by integrating the features of parallel computing and artificial intelligence (AI) [15]. Additionally, Turing can calculate complex tasks, which can be computed by any computer or supercomputer [16]. However, there is a need for the parallel Turing machine to increase the efficiency of given models and algorithms. Thus, the model is trained using the suggested PTIP paradigm to determine the supporting algorithm, and it is then blended to increase the prediction accuracy and classifier fitness. The concept, questions, contributions, and anatomy of the article are introduced in the next subsections. Our motivation stems from the fact that we are using data from social networking in conjunction with academic and professional data to enhance predictive analytics. Several cutting-edge algorithms have been studied [17,18] to comprehend the proposed PTIP paradigm. The following general research questions are what this work aims to explore and answer:

- Can matching, fitness, and accuracy ratings be used to fine-tune and combine algorithms?
- Is it possible for a blended model to perform better than a single algorithmic model when it comes to handling bias, over- or under-fitting, poor generalization, and low accuracy?
- Can error be used/programmed to govern the model for lower and upper bounds to avoid bias and over-learning?
- Can algorithms be added and deleted algorithmically in real-time as the classifier processes the provided raw data for ML predictions?
- Can overfitting and underfitting be pushed to a logical space of the optimum fitting?
- Finally, can a blended model learn from its incorrect predictions?

1.1. Research Problem

The main challenge that data scientists, researchers, and analysts come across is to know which algorithm or model is the best for the problem(s)/data to be used? There are many unknowns and uncertainties. Each algorithm may behave differently for different dataset and variables/features. Evaluation of a model or algorithm is crucial for various classifiers. It becomes highly challenging to choose a classifier type based on new or unseen data. The standard rule adapted by the scientists and researchers is assessing some of the classifiers that algorithm produces [19]. We consider this as a research problem to see if an improved model with algorithms, uniquely engineered with built-in parallel processing, can be introduced for such predictive modeling using enhanced blending and tuning.

1.2. Research Contributions

We see the potential of data science in the creation of new algorithms as well as enhancements to current strategies, while considering the most recent requirements and applications in the big data environment. The following is a summary of the paper's contributions:

- The proposed PTIP has the ability to measure the accuracy, fitness, and matching scores in order to create a blended process;
- The proposed approach demonstrates that blending is performed depending on how well each algorithm, as chosen from the pool of options, fits the sort of data model being trained. Furthermore, the proposed approach guarantees that the model is neither biased nor overfit in comparison to any particular algorithm;
- The proposed classifier is trained to be able to add a good-fit algorithm (one with better metrics) or get rid of a bad-fit algorithm (one with inferior metrics);

- The enhanced metrics, which control the model's performance dimension, are produced. In this manner, the proposed approach is simultaneously cross-checked while learning from the data (during training);
- The proposed model's design is represented in 3-D logical space. In addition, the proposed model is shifted to z-space coordinates to get the optimal fitness;
- A novel strategy to using a meaningful measure in machine learning is created (i.e., Error). The presented model has been trained to be managed by minimum and maximum error-bounds. As a result, acceptable bias and fitness, including overlearning, may be preserved. The lowest and maximum error boundaries for GG and LG are 20% and 80%, respectively.

1.3. Paper Structure

The remainder of the paper is structured into the following sections:

Section 2 presents the salient features of the current studies. Section 3 outlines the proposed parallel turning integration paradigm. Section 4 presents the experimental findings and discussion. In Section 5, the entire article is concluded.

2. Related Work

This section summarizes the most important aspects of existing approaches. Ref. [20] described an iterative approach for continuous variational methods that use the boundary value issues. More specifically, they investigated a parallelization technique that makes use of the capability of multiple cores GPUs (graphics cards) and CPU (Central processing unit). They also investigated the parallel technique for first- and second-order Lagrangians and demonstrated its superior performance in two intriguing applications: a fuel-optimal navigation issue, known as Zermelo's navigation problem, and an extrapolation problem. The cross-modal Turing test was used as an all-purpose communication interface that enabled an embodied cognitive agent's "message" and "medium" to be tested against another [21]. Two competing modules might undergo a sequential cross-modal Turing test with the usage of reciprocating environments and systems. Such a strategy could prove to be crucial in systems that require rapid learning and model adjustment, (e.g., cyber-physical systems), which are formed at the intersection of many technical–scientific engineering solutions. This strategy may be successful in the field of transfer learning, where a pre-trained network fragment might well be linked with input that is essentially unrelated to neural networks.

Researchers looked into potential increases in sampling efficiency [22]. To depict computationally ubiquitous models, a polynomial-time Turing machine was utilized, and Boolean circuits were used to represent artificial neural networks (ANNs) functioning on finite-precision digits. Direct connections between their inquiry and the outcomes of computational complexity were shown by their study. On the anticipated increases in sample efficiency, they offered lower and upper bounds. The input bit size of the required Boolean function defined their bounds. They also emphasized the close connections between the intensity of these limitations and the standard open Circuit Complexity concerns. Deep learning in machine learning seeks a general-purpose computing device that could carry out challenging algorithms resembling those in the human brain. Neural Turing Machine (NTM) integrated a Turing machine with a long-term memory, which was used as a controller to accomplish deep learning techniques [23]. Basic controllers are used by NTM to carry out a variety of simple and sophisticated tasks, including sort, copy, N-gram, etc. Complex jobs, such as classifications, were disregarded, and NTM's weights could not be improved.

Super-Turing models' expressiveness hierarchy was introduced [24]. A-decidable and i-decidable algorithms (e.g., D-complete, U-complete, and H-complete) complexity classes—inspired by NP-complete and PSPACE-complete classes for intractable problems—were introduced as first steps. Similar to how to approximate, randomized and parallel algorithms enabled workable solutions for intractable problems. This study might be seen as the first step

toward feasible approximations of answers to the Turing machine's inherently unknowable issues. All of the existing methods focused on decision making and model adjustment. Contrary, the proposed PTIP is able to quantify the matching, fitness, and accuracy scores to establish a tuning and blended process. The proposed PTIP also ensures that the model is neither overfit nor biased, as compared to any single algorithm under test. Furthermore, a distinctive approach of employing a significant metric in machine learning (i.e., Error) is developed. Finally, the proposed model is trained to be administered by minimum and maximum error bounds. Thus, acceptable bias and fitness including overlearning can be maintained. The contemporary nature of existing approaches is given in Table 1.

Table 1. Contemporary contributions of existing studies.

Approaches	Turing Operations for Machine Learning	Features	Vulnerabilities
IMDV [20]	Iterative approach for continuous variational methods	A parallel approach that employs the capabilities of several cores of GPUs (graphics cards) and CPUs (Central processing unit) for addressing the extrapolation problem	Its performance is limitted to two applications:a fuel-optimal and navigation problem.
CMTT [21]	Cross-modal Turing testing process	The cross-modal Turing test was employed as an all-purpose communication interface for the rapid learning and model adjustment	This method could only work for transfer learning.
PTTM [22]	Utilization of the polynomial-time Turing machine	Projected increases in sample efficiency, offered lower and upper bounds between the intensity of these limitations	Increased circuit complexity
NTM [23]	Neural Turing Machine with a long-term memory,	Neural Turing Machine is integrated as a Turing machine with a long-term memory, to carry out variety of simple and sophisticated tasks, including sort, copy, N-gram, etc. Complex jobs, such as classifications	The weights of neural Turing machine could not be improved
AIA [24]	Introduction of AIA algorithm based on super-Turing models	Approximate, randomised, and parallel algorithms were used to solve intractable problems	Inaccurate for decision-making
Proposed PTIP	Parallel Turing integration paradigm	Proposed PTIP is able to quantify the matching, fitness, and accuracy. Furthermore, the proposed PTIP ensures that the model is neither overfit nor biased	Trained to be administered by minimum and maximum error bounds

3. Proposed Parallel Turing Integration Paradigm

The latest trend in the ML research [25] has shown an immense potential to evaluate various algorithms in parallel [26]. We take a research opportunity to enhance the accuracy for the state-of-the-art algorithms of the ML. The proposed PTIP consists of four components to play an important role for accuracy enhancement of the ML algorithms.

- Improved Machine Learning Engine
- Develovement of Pre-processing Internals
- Measurement and Risk Optimization
- Final framework of PTIP

3.1. Improved Machine Learning Engine

The proposed PTIP leverages the features of an improved machine learning engine (IMLE) that increases accuracy depicted in Figure 1. The IMLE consists of state-of-the-art components, which help guarantee the QoS provision, especially accuracy. The PTIP is discussed at the abstract level.

- Enhanced Algorithms Blend and Turing (EABT): The EABT uses ensemble, bagging, and boosting to choose the best blend until the improved metrics are evaluated, then it applies the current ML and predictive modelling techniques (e.g., Logistic Regression, Linear Regression, Multiple Regression, Bayesian, Decision trees, SVM, and Classification). This scheme exploits the potential of parallel processing of the techniques. Errors and bad predictions made during a test are sent back to the algorithm in this phase so it can continue to learn from its errors (AI).
- Enhanced Feature Engineering and Selection (EFES): It improves feature engineering to extract more information (e.g., feature development and transformation). Finding the best collection of characteristics or features for the relationship between "predictor-target" variables is the best fit. Additionally, it makes sure to include features with the greatest fitness scores and eliminate features with the lowest. Each feature may be validated by the EFES.
- Enhanced Weighted Performance Score (EWPM): It ensures that the model should neither be overfit nor underfit by generating a unique metric based on standard metrics. It also acts as the measuring instrument to ensure the precision of overall performance metrics.
- Improved Cross Confirmation and Split (ICCS): This component develops and employs a novel strategy to train-test data splitting in order to improve existing validation approaches such as cross-validation.

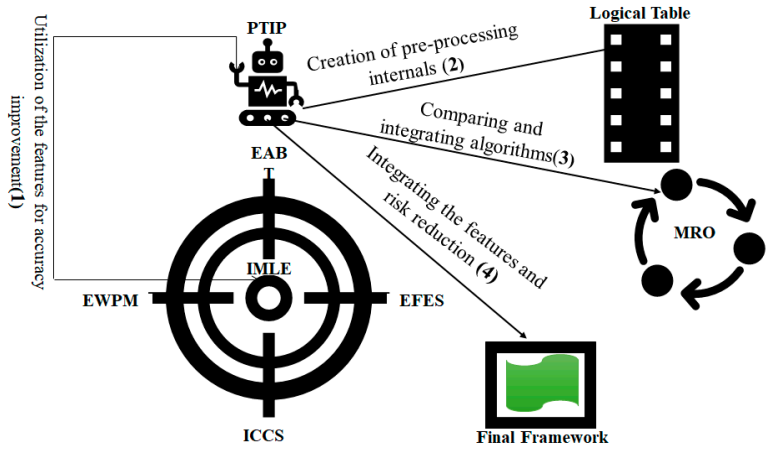


Figure 1. Functional components of PTIP for accuracy enhancement.

3.2. Development of Pre-Processing Internals

The pre-processing internals reduce the inaccuracies in the datasets. Therefore, the pre-processing internals can be created using the Logical Table (LT). The complete modular details of the Logical Table (LT) are outside the scope of this article. However, the brief detail is provided. The method that best fits the PTIP is to use of the logical table, which expands with the quantized output. We determine that the proposed model's minimum value is where LT controls the process. According to Algorithm 1, it makes an entry for each dimension (X, Y, and Z). The threshold value is set for the LT; if the LT value < 0.5, binary "0" is assigned; if ≥ 0.5, binary "1" is assigned. Thus, the binary truth table can be formed from those values. The primary goal of Algorithm 1 is to employ LT to produce pre-processing internals in order to correct the inaccuracies in datasets.

Steps 1–2 of Algorithm 1 depict the input and output procedures. The variables utilized in the Algorithm 1 are initialized in steps 3–5. The logical table objects and nodes are created in step 6. The IMLE is generated in step 7 to increase accuracy. Step 8 configures the IMLE's component (*FFES*) (which helps to increase accuracy). The value for error removal in the IMLE is shown in steps 9 through 12, and the error removal procedure continues until no more errors are found, in every direction. Step 13 involves the error and data validation processes.

Step 14 provides the distance function's correct arguments. Steps 15–16 show the activation processes of Turing and blended functions, which provide the alert on the logical table. If an error is discovered, the error classifier functions of $\mathbb{B}_A$, $\mathbb{T}_A$ are invoked and the best set of features is determined in steps 17 through 19. The alert is set on the logical table to identify errors in tuples and blocks in step 20. Additionally, steps 21–22 establish the ideal fitness range. The suggested PTIP has an updated logical table. Step 23 involves setting the strategy function for each dimension. The classifier sets and triggers the cost function for error and data validation. If a value of error < 0.5 is found inaccuracy removal is commenced, and the logical table is eventually modified in steps 24–30.

Algorithm 1: Inaccuracy removal $\psi(n)$ in logical table for development of pre-processing internals

1. **Input:** $A.P = \{A_1, A_2, A_3, \ldots\ldots.., A_n\}$, Raw Dataset $DS(sig, noi)$
2. **Output:** $(\psi(n))$
3. **Initialization(A)**: $\{\ \mathbb{S}(x.y, z)$: strategy function for all dimensions; Op.F: optimal fitness; C: cost function$\}$
4. **Initialization(B)**:$\{$IMLE, EFES$\}$
5. **Initialization(C):** $\{$A: error, A_n: total number of errors; $H_i(p_i)$: entire datasets; $H_i(p_i{}')$: removal of data; $\nabla d(A_1, A_2)$: distance between two data items; $\varnothing(D, t)$: arguments of distance function; $\mathbb{B}_A$: blended function for algorithm; $\mathbb{T}_A$: Turing function; $LT^{\forall}$: alert on logical table;
6. **Generate** LT object, Nodes
7. **Generate** IMLE
8. **Set** FFES, EWPM, ICCS, EAB
9. **While** $(\psi(n))\langle 0.5 \mid (e, E \rightarrow$ IMLE$)$ **Do**
10. **Create** Logical Table Object $LT(x, y, z)$
11. **For each** $(A \in A_n)$ **Do**
12. **Compute** $\nabla d(A_1, A_2) = \sqrt{\left(\sum_{i=1}^{M}\left(H_i(p_i) - (H_i(p_i{}'))^2\right)\right)}$
13. **Validate** $(A \in A_n)$ and $\nabla d(A_1, A_2)$
14. **Compute** $\varnothing(D, t)$
15. **Compute** LT. $\mathbb{B}_A$, $\mathbb{T}_A$
16. **If** $LT^{\forall}\mathbb{B}_A$, $\mathbb{T}_A$ **then**
17. **Retrain** classifier $(\mathbb{B}_A, \mathbb{T}_A)$
18. **Update** EFES $\rightarrow \varnothing(D, t)$
19. **End If**
20. **Compute** Op.F$\{0:1\}$
21. **End For**
22. **Update:** LT of PTIP
23. **Set** $\mathbb{S}(x.y, z) = \sum_{j=1}^{N} \varnothing(D_j, \tau(p_j, S'_{x,y,z})$
24. **Set** C
25. **If** (C is flagged by Classifier) **then**
26. **Compute**$\nabla d(A_1, A_2)$
27. **Detect** $\psi(n)$
28. Update LT
29. **End If**
30. **End While**

LT operates in the memory and is dynamically updated. It keeps track of the algorithms $A(x, y, z) = \{A_1, A_2, \ldots\ldots\ldots, A_n,\}$ as the ML process evolves to accomplish the final optimum fitting after it has incorporated all the algorithms from the pool. This helps achieve the optimum blending and tuning. This logical table stores data based on three dimensions, where 'x' = overfitness, 'y' = underfitness and 'z' = optimum-fitness.

3.3. Measurement and Risk Optimization

The measurement and risk optimization (MRO) component deals with applying various algorithms one by one to observe the outcomes (i.e., measures), and then prepares the model for risk estimation and algorithm blending. This construct also compares the two algorithms at the same time and then groups them based on Euclidean distance for similarity scores in terms of fitness. This sub-model finally produces the set of algorithms for the best fit for a given dataset. The end goal is to engineer inaccuracy removal that can be done through classifier function.

Let us define that standard distance function, which consists of $\nabla d(A_1,\ A_2)$ that demonstrate the distance between two data items. The $H_i(p_i)$ can be identified as entire datasets, and $H_i(p_i{}')$ be the removal of data from the datasets.

$$\nabla d(A_1,\ A_2) = \sqrt{\left(\sum_{i=1}^{M}\left(H_i(p_i) - (H_i(p_i{}'))^2\right)\right)} \tag{1}$$

Let S' be the strategy function for all dimensions, and p_i is the training problem set (dataset vector) in distribution of time $\varnothing(D,\ t)$ being the arguments for distance function in $\varnothing$.

$$\mathbb{S}(x.y,z) = \sum_{j=1}^{N} \varnothing(D_j,\ \tau(p_j,\ S'_{x,y,z}) \tag{2}$$

Therefore, the correct arguments of distance function can be calculated as:

$$\begin{cases} \varnothing(D,\ t) = \frac{\overline{T}}{(D+1)} & t = \infty \\ \frac{t}{(D+1)} & e\ \in N, \quad \overline{T}\ \geq T \end{cases} \tag{3}$$

Theorem 1. *Algorithm evaluation process* $\left\{A_{(1,\ n)}\right\}$ *makes suitability scores for the fitness.*

Proof. Let $\nabla d(A_1,\ A_2)$ be the Euclidean distance between the two algorithms since there must be a matching factor (M.F) for the best fitness between them. Thus, $Op.F\{0:1\} \geq 0.5$, as optimum Fitness scores and $0.2\langle err|Err\{0:1\} < 0.8$, are bounds of LE (err) in GE (Err). With the pointers x denoting Overfitness, y denoting underfitness, and z denoting optimal fitness, let $\mathbb{S}(x.y,z)$ compute the appropriateness scores for the fitness of the provided method in 3-D space. Let $\psi(n)$ be the classifier function, which the model learns to categorize in order to be able to identify the ideal combination of methods for a particular dataset and issue. □

3-D Cube Logic for Security Provision

In encryption transmission, the 3D cube logic is used to maintain a security, and the activity is carried out based on the pattern generated by the transmitting side rather than supplying the whole cube structure. A key that is utilized for encryption and decryption is deduced through a sequence of steps that match the 3D CUBE pattern.

The proposed technique decrypts the encrypted text provided by the receiver using a key derived by completing the shuffling of 3D CUBE based on artificial intelligence (AI), which overcomes the challenge of secured key exchange by removing the key exchange

procedure from the conventional symmetric key encryption. Because it is impossible to deduce a key used by a third party using AI-based learning, the 3D CUBE logic can also provide higher security than the conventional symmetric key encryption scheme.

As it is used to handle keys and transmit them correctly in the symmetric key encryption system, the procedure after generating the key in the 3D CUBE technique is identical to that in that mechanism. In the symmetric encryption system, instead of participants sharing keys used during initial encryption communication, the transmitter sends a 3D element key pattern as a randomized 3D CUBE sequence by which the key can be activated, and the recipient obtains the secret key by incorporating it through AI-based learning. By using the XOR operation on the sequence's 3D CUBE pattern, the secret key is formed here based on the randomized order as a sequence.

Assume a dataset $Des(\text{sig}, \text{noi})$, where "sig" denotes the signal and "noi" denotes the noisy component of the dataset that can be transmitted. The classifier function with a loss function $\text{L(x, y, z)}|\ (0:1)$, for which we loop in n-sample blocks such that loss function remains in the defined boundary as estimated, for which the feature sets exist in the function $F = \{f_1,\ f_2,\ f_3, \ldots\ldots.., f_n$ with optimal score > 0.5. The classifier for the upper boundaries of generalization error in the probability $\acute{C}(x, y, z)$, for n blocks of data sample can be provided by:

$$\acute{C}(x, y, z) = L(x) \leq L(y) \leq L(z) + \frac{1}{n} \sum_{k=1}^{n} (U.Des - L.Des)^k + \sqrt[3]{\frac{-\log(\frac{p}{2})}{2\varphi}} \tag{4}$$

where φ is the pattern identified as signals (removing noises) with the probability of $1 - \text{p}$. Here, we create a straightforward approach to calculate the loss function in noisy and signal data(N) that affects classifier construction.

$$L(DS(S(x, y, z), N(x, y, z) = \begin{cases} 0, & (N(x, y, z) \geq 0.5\ \geq S(x, y, z)) \\ 1, & (S(x, y, z) \geq 0.5\ \geq N(x, y, z)) \end{cases} \tag{5}$$

Thus, for each algorithm in the pool, the optimum fitness can simply be determined as:

$$Op.F\{0:1\} = ||(\acute{C}(x, y, z) - L(DS(S(x, y, z), N(x, y, z))||^2 \times log\frac{(err)^2}{(err + Err)} \tag{6}$$

Based on the Equations (4)–(6), the optimum fitness values can be calculated, which are given in Table 2.

Table 2. Quantified comparison of optimal fitness.

Op.F(x)	**Op.F(x,y)**	**Op.F(x,y,z)**
0.00028723	0.00238712	0.03489120
0.234356	0.93	0.87
0.659023	+0.73	+0.78

As previously mentioned, our model is built on 3-D space for the x, y, and z dimensions, which the algorithm employs to optimize the blend's fitness (Op.F) to the supplied data. It is important to highlight that using this technique, the model is designed to be highly generalizable for any supplied data with any kind of attributes. We thus engineer the mix utilizing matrix (real-valued) space manipulation and the Frobenius norm provided by:

$$||M||(x, y, z) = \sqrt{\sum_{x=1}^{X} \sum_{y=1}^{Y} \sum_{z=1}^{z} M_{x,y,z}^2} \tag{7}$$

The matrix manipulation for each dimension to evaluate the blend can be built as:

$$\psi(n) = \underset{(x,y,z\ \in Z)}{Extract}\,(||M||(x,y,z)) + \log\sum_{k=1}^{Z}\gamma^{z+1}\,.\,Op.F\{0:1\} \tag{8}$$

The illustration in Figure 2 supports the mechanics of the Theorem 1. As we observe the visualization of three functions, err | Err (e/E), Op.F and Cost (C) being rotated based on distance function, as shown. Thus, Blender Filter Switch (logical) connect the value to the Suitability and Evaluation Score 3-D logical construct. The value of $\psi(n)$ in each dimension swing between 0 and 1, based on Op.F response from Tuning Synthesizer block. Table 3 shows the results of developed functions.

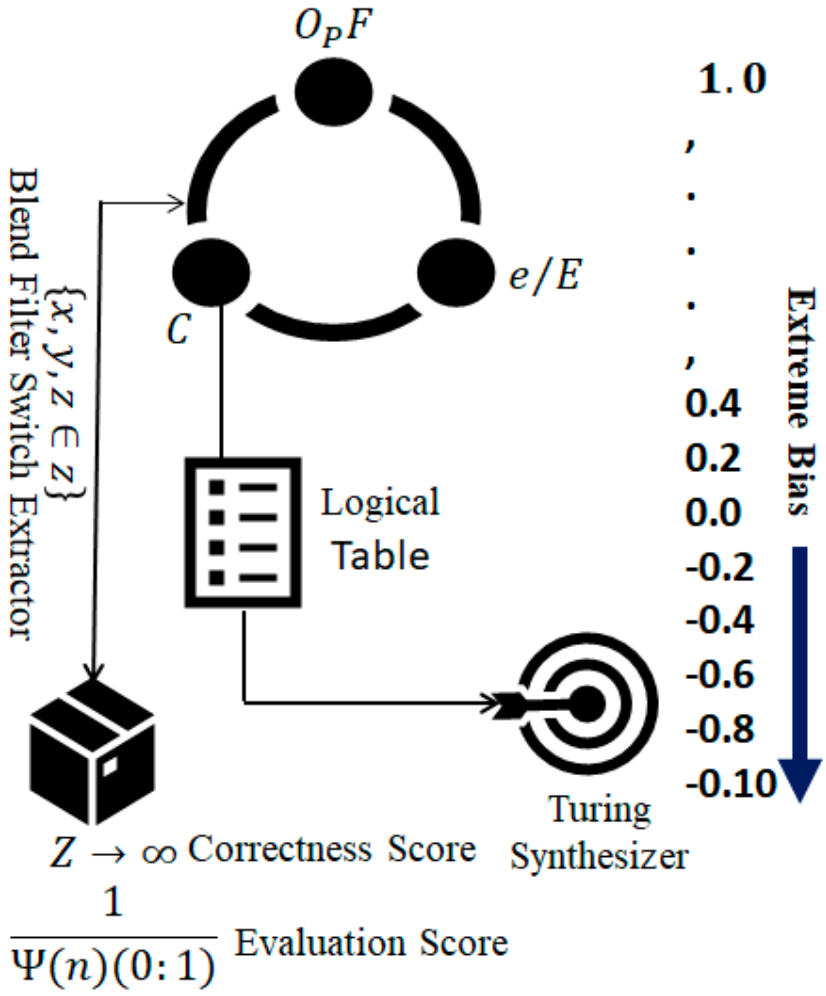


Figure 2. Illustration of Internals of the PTIP for preprocessing.

Table 3. Shows the results of ten experiments performed on developed functions.

Function	1	2	3	4	5	6	7	8	9	10
$\psi(x)$	0.0141	0.0132	0.0296	0.0910	0.1384	0.2780	0.4692	0.5901	0.6220	0.6793
$\psi(y)$	0.9301	0.9100	0.0410	0.8323	0.7928	0.6029	0.5921	0.4301	0.3120	0.2872
$\psi(z)$	0.0130	0.0570	0.0962	0.1502	0.9991	0.4170	0.4288	0.5919	0.6129	0.7709
$\psi(x,y,z)min$	0.0021	0.0091	0.0101	0.1303	0.2920	0.3409	0.3110	0.3300	0.3419	0.3910
$\psi(x,y,z)max$	0.5412	0.5900	0.6001	0.6400	0.6408	0.6604	0.6923	0.7129	0.7021	0.7400

The red circled values should be considered as errors that require further investigation.

Theorem 2. *Risk Estimation, Local Errors, and Metrics Evaluation—*$Algo.Risk_{emp}[f]$
The algorithm generalization evaluation error {AGE(Err)} is inbound of all LE err(n), for which each occurrence of the error at any point in x and y space, exists inside all theoretical values of Err, such that $err(A) \in Err(A+1)$, *where* $0.8 < e < 0.2$. *Let there be a maximum risk function (Φ) with mean square error* (MSE)*on the set of features as* $F = \{f_1,\ f_2,\ f_3,\ \ldots\ldots,\ f_n\}$, *and unknown means as* $\{m_1,\ m_2,\ m_3,\ \ldots\ldots,\ m_n\}$.
The algorithm generalization assessment error, or AGE(Err), is an aggregate of all LE err(n), in which each instance of the error at any location in x and y space occurs within all conceivable values of Err, so that $err(A) \in Err(A+1)$, *where* $0.2 < e < 0.8$. *Assume that the maximum risk function (Φ) has unknown means (m 1, m 2, m 3, m n) and mean square error (MSE) on the set of features (F= f 1, f 2, f 3, . . . , f n).*

Construction. To prevent underfitting and overfitting in terms of the errors to be regulated by the upper and lower bounds. We construct the minimum and maximum error boundaries logical limit. We initially force the mistake to be at a low threshold before making it high. The algorithm then learns to keep between the min(e:0.2) and max(e:0.8) boundaries, and accuracy is obtained.

$$max(e:0.8) = \frac{1}{E}\sum_{i=1}^{Ne}\{(RMSE_i) - (100+0.2)/E\} \tag{9}$$

$$min(e:0.2) = \frac{1}{E}\sum_{i=1}^{Ne}\{(RMSE_i) - (100+0.8)/E\} \tag{10}$$

Let us define our optimum error (dynamically governed by algorithm tuning and blending process), as:

$$E_{opt}\ (max,\ min\ |e_{(x,y,z)} = \left(\frac{\int \partial(x,y,z)}{(1-E)}\right) \times \sum_{j=1}^{Nt}\left(\max(e)_j - \min(e)_j\right)\ *\ g.f \tag{11}$$

where the PTIP algorithm generates (g.f), the error gain factor. Each method tends to cause more mistakes when they are combined, hence the combined errors of both global and local functions must stay within the given range. There are two types of errors for generic machine learning modelling: estimation and approximation [27]. Its collective name is generalization error, and finding a specific function f′(x, y) that tends to reduce the risk of training in the targeted space is what we aim to do in this case (i.e., X, Y, Z), shown as:

$$Risk[f']_{x,y,z} = \int_{X\times Y\times Z} L\ (y,\ \ f'(x,y))\ P(x,y,z)dxdydz \tag{12}$$

Figure 3 shows the concept of correlation of error bounds and risk function. As we propose the novel idea of limiting the error between 20% and 80% for optimum realistic fitness for the real-world prediction, it shows that the Risk Estimation (emp) stays in bounds of logical cube shown on the right. The center point shows the ideal co-variance of function z_i. P (x, Y, Z) will be unknowable at this time. Based on the "empirical risk minimization principle", which is statistical learning theory that can employed to determine the risk.

$$Risk_{emp}[f'] = \frac{1}{m}\sum_{i=1}^{m} L\ (\ y^i\ ,\ f'\ \left(x^i\right)) \tag{13}$$

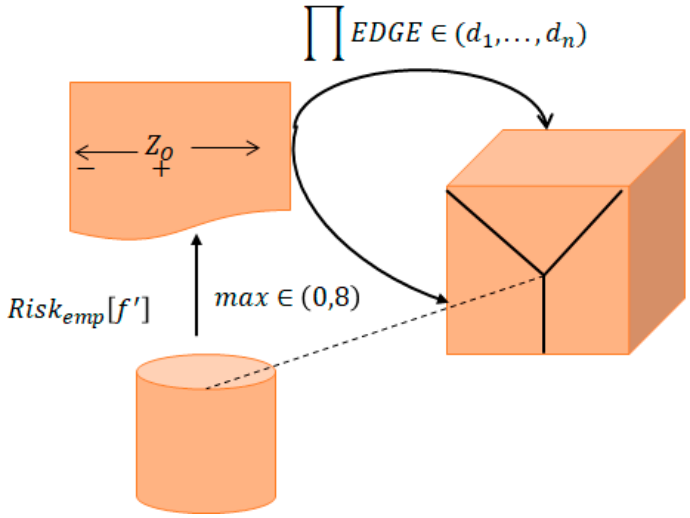


Figure 3. Illustration of maximum error bounds for risk estimation global function.

Here, we require to fulfill two conditions, which are given below:

(i) $\lim_{m \to \infty} Risk_{emp}[f'] = Risk[f']_{x,y,z}$

(ii) $\lim_{m \to \infty} \min_{f \in H} Risk_{emp}[f'] = \min_{f \in H} Risk[f]_{x,y,z}$.

These two conditions could be binding when H is comparatively trivial. The second condition needs minimal convergence. Thus, following bound can be constructed that is being held valid with probability of $1 - \delta$ is $R[f] \leq Risk_{emp}[f'] + z_{\epsilon}$. Thus, z_{ϵ} can be determined as:

$$z_{\epsilon} = \sqrt{\left(\frac{h\, ln\left(\frac{2m}{h} + 1\right) - ln\left(\frac{\delta}{4}\right)}{m} \right)} \tag{14}$$

The sub-estimator function is $\hat{m}_k = c(F_k, \vartheta)$. Hence, the regularization parameter ϑ is positive and it has been observed that the sub-estimator function $c(F_k, \vartheta) = F$ such that, $\vartheta = 0 \mid \hat{m}_i = F_k$. We deduce that $\vartheta = \infty$, corresponds to maximal shrinking, which is $\{\hat{m}_k = 0,\ for\ k = 1,,,,,,n\}$. In this case we use Stein's unbiased risk estimate (SURE) and cross validation (CV) technique, where prominent estimators include (lasso), (ridge) and (pretest). We use the squared error loss function, sometimes called compound loss, as the basis for loss and risk estimation.

$$Loss_n(F,\ c(k, \vartheta),\ D) = \frac{1}{n} \sum_{k=1}^{n} (c(F_k,\ \vartheta\ ) - m_k)^2 \tag{15}$$

where $D = \{d_1, d_2, \ldots\ldots\ldots, d_n\}$ shows the distribution of Features F {p, q, r}. It should be observed that Loss is highly dependent on 'D' through value of 'm'. We can construct the regularization parameter for which the algorithm blend fits the model to maximum relevance, such that: $\vartheta(D) = \max_{\vartheta \in [0,\infty)} B\ (c(.\ ,\ \vartheta),\ \pi)$.

Therefore, the risk of algorithm $\left(Algo.Risk_{emp}[f]\right)$ can be calculated as:

$$Algo.Risk_{emp}[f] = Risk[f']_{x,y,z} \times \prod_{EDGE(Loss_n)} z_{\epsilon} n \tag{16}$$

Consequently, Equations (15) and (16) can be used to determine the maximum risk of algorithm Φ.

$$\Phi = \max\left(Algo.Risk_{emp}[f]\right) + \lim_{0.2 < err < 0.8} Risk[f']_{x,y,z} \tag{17}$$

where $Risk[f']_{x,y,z}$ is the risk of distribution features of algorithm.

The risk estimation process of algorithm in 3-d space is depicted in Figure 4 to support in-bound local evaluation (LE) and global evaluation (GE).

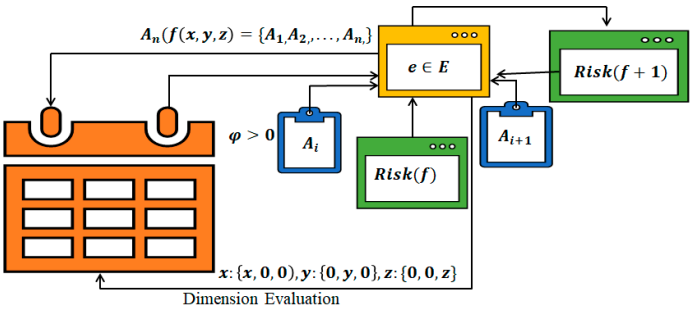


Figure 4. Illustration of risk estimation function in 3-d space for in-bound LE and GE.

Algorithm 2 maximizes fitness, this algorithm computes the risk estimation function, local error bounds, and metrics evaluation. It builds on Theorem 1. It tunes the blend to stay within error bounds for ensuring optimum fitness based on the Risk Estimation Function and observed errors, and then assesses metrics to ensure the blend's overall performance.

Algorithm 2: Computation of risk estimation process

Input: $\psi(n)$: classifier Function, F{n} : Features set for sampling
Output: Φ, $\text{Risk}_{\text{emp}}[f]$, e, E, Ë
Initialization: $e \leftarrow 0$, $E \leftarrow 0$

1. **Create**: LTObject, $\text{NODES}_{i \in N}$, PTIP
2. **Generate** ObjectIMLE(h)/* Generate an object reference for improved machine learning engine application programming interface (API)*/
3. **Set** ObjectIMLE.PublicFunctions(h.IMLE, h.EFES, h.EWPM, h.ICCS)/* managing the four constructs*/
4. **While** (e⟨0.2 OR⟩0.8)**Do**
5. **Compute** max(e) and min(e) bounds
6. **Compute** $\sum_{j=1}^{Nt} \left(\max(e)_j - \min(e)_j\right) * g.f$ /* The optimum error function */
7. **Create** Logical Table Object LT(x, y, z)
8. **Create** $h.ICCS(S_1, S_2, , , , , , S_n)$ /* Create Test Train Split using ICCS library. */
9. **For** (Each node in $\text{NODES}_{i \in N}$) **Do**
10. **Compute** $\text{Risk}_{\text{emp}}[f'] \leftarrow \frac{1}{m} \sum_{i=1}^{m} L(y^i, f'(x^i))$
11. **Compute** Loss Function $\leftarrow \frac{1}{n} \sum_{k=1}^{n} (c(F_k, \vartheta) - m_k)^2$
12. **If** $(\vartheta(D) \geq$ Last updated $\max_{\vartheta \in [0,\infty)} B(c(.,\vartheta), \pi))$ **then**
13. **Replace** $\vartheta(D)$
14. **Recompute** Loss Function
15. **End If**
16. **Compute** z_i
17. **Set** $R[f] \leq \text{Risk}_{\text{emp}}[f'] + z_i$
18. **Examine** error bounds for regularization
19. **If** (error > 0.2 and < 0.8) **Then**
20. **Update** the LTobject.LT(x, y, z)
21. **Update** x, y, z from Logical Table
22. **End If**
23. **End For**
24. **Compute** $(\text{Algo.Risk}_{\text{emp}}[f], LT)$
25. **Compute** Φ
26. **Compute** e, E and E_{opt} /*Using Risk Function and LTobject Update *
27. **End While**
28. **Return** Φ, $\text{Risk}_{\text{emp}}[f]$, e, E, E_{opt}

Theorem 3. *$Algo.Blend(p,q,r)$, $Algo.GG(\mathbb{G}(p,q,r)$, and $Algo.Error(p,q,r)$ support the parallel Turing integration paradigm.*

Proof. Let us take the pools of the supervised learning algorithms such as $\mathrm{A(p,q,r)} = \{\mathrm{A_1, A_2, \ldots\ldots\ldots, A_n}\}$, which acquires the given data $D_s = \{\mathrm{ds_1,\ ds_1,\ ds_3, \ldots\ldots\ldots, ds_n}\}$; thus, the correlation exists. Therefore, matching factor M_f can be expressed as M_f $(\mathrm{p} \to 0, \mathrm{q} \to 0, \mathrm{r} \to \infty)$. The LG exists for each algorithm such that GG is available in optimized fashion that is illustrated as $\mathrm{g} \in (1 - \mathbb{G})\ ||(\mathrm{A} \in \{1, 2, \ldots, \mathrm{n})$. The optimization functions exist for each algorithm. Therefore, each algorithm possesses the set of predictor characteristics, such that $\mathrm{S_{input}} = \{1, 2, \ldots, \mathrm{T_n}\}$, and targeted variables, $\mathrm{S_{output}} = \{1, 2, 3, 4, \ldots, \mathrm{T_n}\}$. Thus, each given algorithm A performs well in the blend for each set of features S_{fe}. Hence, the enhanced weighted performance metric W_{pm} should be $\geq$ 0.75 for each specified parameter, which remains above the threshold value of the measurement until W_{pm} falls below 0.75. Therefore, $\mathrm{A} \in$ All in $\mathrm{A_{x,y,z \leftarrow 0.1}}$. The LE becomes very unstable in blend, and thus the model creates a complex error function ($\mathbb{E}$(x, y, z,) such that $\mathrm{e} \in \mathbb{E} \mid \log(\mathbb{E}^{\mathrm{e}} - 1) > 0$, and let $\forall$ be the measure of complexity of the GE within a probability $\mathrm{p} > 1 - \delta$, where all $\delta \subseteq \frac{1-\mathrm{p}}{\mathrm{p}^2}$. There must be a hypothesis "h" on GE that travels in the x, y, and z directions. It shows nonlinearity that is dependent on the number of algorithms in the blend. Moreover, it is distributed linearly throughout the space in the LE. □

Construction. The E_d, H_d, M_d functions can be employed in the space. The optimum GG is used for induction. Therefore, Euclidean distance can be used for measuring the distance that is given by:

$$E_d = \sqrt{(x_1 - x_2)^2 + (y_1 - y_2)^2} \tag{18}$$

Similarity can then be obtained using Hamming distance H_d, which is specified by:

$$H_d = \sum_{j=1}^{k} |x_j - y_j| \tag{19}$$

Thus, Minkowski distance M_d can be calculated as:

$$M_d = \left(\sum_{j=1}^{k} |x_j - y_j|^p \right)^{\frac{1}{p}} \tag{20}$$

We create the related matching factor (MF) based on these data at the point when each distance reduces to the smallest potential value with the least amount of theoretical GE (Err).

$$MF(x,y,z) = \frac{1}{S} \sum_{i=1}^{n} (A_i) \pm \left\{ \frac{(E_d + H_d + M_d)}{n} \right\},\ Err < 0.5 \tag{21}$$

The triangles are built for our blend using the 3-D space and axis align method. Let $\Delta = \{\Delta_1, \Delta_2,\ \Delta_3, \ldots\ldots\ldots, \Delta_n\}$, which specifies the immeasurable number of triangles in disseminated fashion which are given as:

$$\Delta_{(x,y,z)}^{(a,b)} = \{\mathbb{W}(x \leftarrow a\ ,\ y \leftarrow b,\ z \leftarrow\ ab\ :\ a \leq\ b,\ and\ ab > 0\} \tag{22}$$

The following function describes the search for optimal coordinates: where $\mathbb{W} = \begin{cases} 1,\ a \leq\ b \\ 0,\ ab > 0 \end{cases}$ and EWPM is obtainable through IMLE API. Let us assume that EWPM

should remain above 0.5 for optimal zone in the 3D rational space. Thus, the co-ordinates for the y-axis can be expressed as:

$$Y_i = \alpha + \begin{bmatrix} \alpha \\ \beta_1 \\ \beta_n \end{bmatrix}_1 X_{i,1} + \ldots\ldots\ldots \begin{bmatrix} \alpha \\ \beta_1 \\ \beta_n \end{bmatrix}_n X_{i,n} + \epsilon_i, \text{ Where } \epsilon = \begin{bmatrix} \epsilon_1 \\ \epsilon_2 \\ \ldots \\ \ldots \\ \epsilon_n \end{bmatrix} \text{ in each dimension}$$

As a result, given corresponding values and blend of the algorithms switches from multi-dimensional space to lower dimension. After tuning has been accomplished (Theorems 1–3), an error happened as a complicated function, dependent on a variety of variables and the fundamental blend of algorithms. The global err is displayed in Figure 5 as every algorithm's LE is no longer applicable mathematically. The Each algorithm's standard deviation error (σ) tends to rise when it is coupled with another algorithm. A common statistical method that models relationships between n+ variables using a linear equation to determine its fitness is developed by:

$$Standard\ Deviation\ Error = \sigma = \sqrt{\left(\frac{\sum_j \gamma^2{}_j}{(n-p-1)}\right)} \tag{23}$$

$$y_i = f(x) = \begin{cases} 0, & If\ error\ falls\ above\ the\ threshold \\ 1, & If\ error\ is\ acceptable. \end{cases} \tag{24}$$

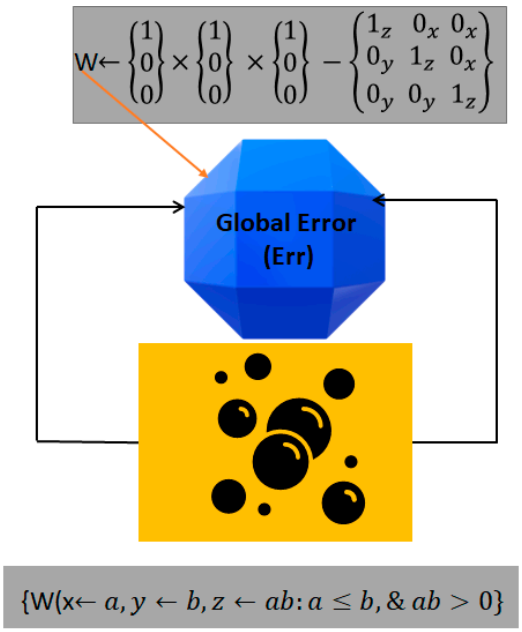


Figure 5. Weight of Global error.

The distribution can be written using Bernoulli Distribution process:

$$Y_i\colon\ Prob\{Y_i = y_i\} = \pi_i^{y_i}(1-\pi_i)^{1-y_i}$$

The bounds of 20% and 80% should be tuned. The small circles demonstrate the (x, y, z) infinite space in the logical cube depicted in Figure 6. Note that each point varies between 0 and 100%. It is ideally used when output is binary, such as 1 or 0, Yes or No,

etc. A logit function governs the linear model in LR. Thus, we can finally construct using Equation (25).

$$Err^2 = \left(\sqrt{\frac{(Err_{Measured} - Err_{Calculated})}{n}} \right)^{n^2} \tag{25}$$

$$H(s) = -p_1 log\ p_1 - p_1 log_1 \ldots\ldots - p_2 log\ p_2 \ldots\ldots - p_n log\ p_n \tag{26}$$

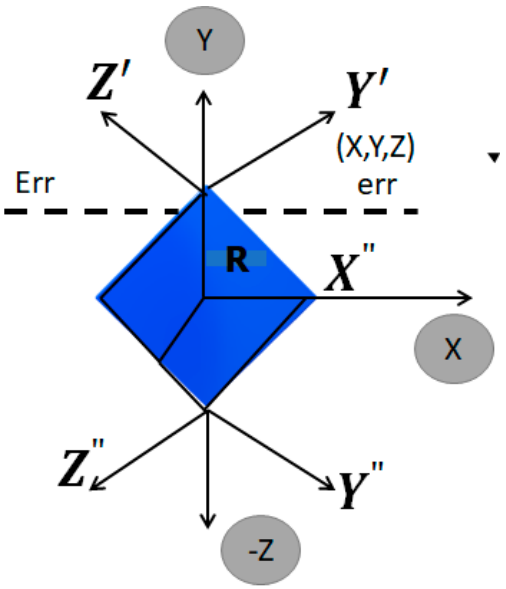


Figure 6. Logical illustration of complex error function in terms of LE and GE spread in each dimension.

Normally we encounter classification error with training error and testing error. If we assume D to be the distribution and 'exp' is an example from D, and let us define target function to be f(exp) indicating the true labeling of each example, exp. During the experiment of training, we give the set of examples as $exp_1, exp_2, exp_3, exp_4 \ldots\ldots exp_n$ and labelled as $l(exp_{1)}, l(exp_2), l(exp_3), l(exp_4) \ldots\ldots l(exp_n)$.

We examine for incorrectly classified examples with the likelihood of failure in distribution D in order to identify the overfitting.

Then, for each algorithm, we build a gain function that is dependent on fundamental performance metrics including bias, underfitting, overfitting, accuracy, speed and error [28]. We only consider bias-ness (B), underfitness (UF), and overfitness (OF) as factors that influence how the fundamental algorithm learns. As a result, we may start creating the gain function $\mathbb{G} = gain\ \{A\ \in (1, 2, 3, 4, \ldots\ldots,\ n)\}$. As illustrated in Figure 7, we establish the GG to ensure that the LG for every blend reduces the distance between the two triangles in order to create the blend function. Furthermore, Figure 7 also illustrates over-probability stretch using three dimensions. The correlation of the features g, gain function and over probability ρ can be determined as:

$$0 \leq g(a, b, c) \leq \sum_{i=0}^{n} \binom{a, b, c}{C} \leq \rho \tag{27}$$

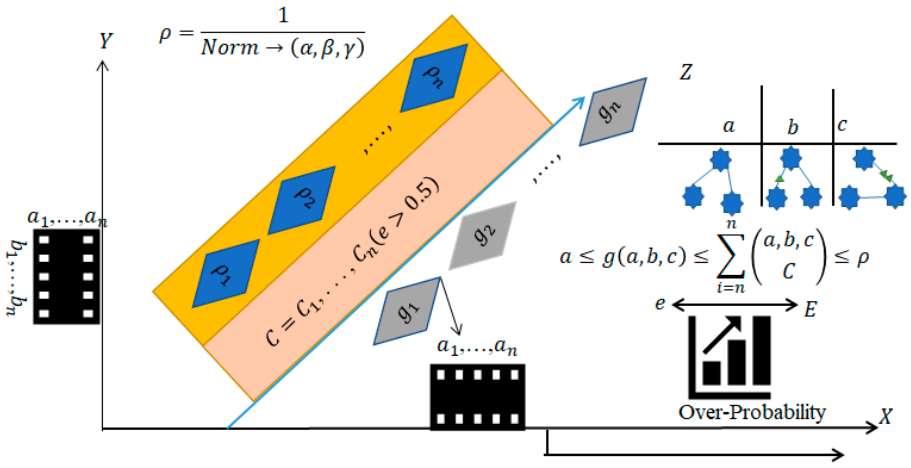


Figure 7. Blend minimization distance between two triangles using over-probability.

Overfitness (z) is observed to be reduced in the x and y coordinates. The primary purpose of GG is to maintain the point in the area where the mistake "err" is acceptable.

We tweak the gain function to ensure that in the final blend we may filter the algorithm and progressively blend to be as optimal as is feasible. This is done in considering the fact that LG will be extremely high in lower dimension, and GG designated as $\mathbb{G}$ will be lower in lower dimension. As rationally demonstrated in the algorithm formulation, the approximation function (AF) connects the score factor between a collection of predictor characteristics S_{input} = {1, 2, 3, 4,....., T_n} and target variables S_{output} = {1, 2, 3, 4,....., T_n}. As a result, we compute GG by employing Equations (24)–(28).

$$\mathbb{G} = \frac{(LF_1 - LF_2)}{Norm \rightarrow (\alpha, \beta, \gamma)} \times e^n \frac{1}{2} \sum_{i=1}^{m} \sum_{j=1}^{m} \alpha\beta\gamma_{i,j} < \alpha_{i,j}, \beta_{i,j}, \gamma_{i,j} > + \sum_{k=1}^{K} ((\alpha\beta\gamma)_k) \quad (28)$$

3.4. Final Framework of PTIP

A robust framework that combines hardware, firmware and software optimization is needed to optimize an ML space and algorithm. Several cutting-edge algorithms are utilized to support the software optimization tactics. The final framework is supported by the regularized risk reduction approach and kernel learning in different segments.

Let $\Omega(*, x|y|z)$ be a convex function for unconstrained minimization of the learning the regularized risk minimization, where space is considered distributed in 3-D dimension, for which our goal is to optimize the algorithm blend in z-dimension with minimum cost function $\mathbb{C}(a, b, c)$, such that the averaged empirical risk function (Φ(t)) associated with algorithm blend inside the boundaries of convex function. Thus, we construct:

$$\Omega(*, a|b|c) = \Omega(*, a|b|c) \in \left(\mathbb{F}.\rho \left\{ \frac{f(A, B, C)}{(x - xy)^2 \times (y - z)} \right\} \right) \quad (29)$$

To conduct the experiments, typical experimental values for the local gain, global gain, and cost function have been chosen which are provided in Table 4.

Table 4. Typical experimental values during training.

	Min	Max	Averaged	Out of Bounds
$LG\{g(a, b, c)\}$	0.21	0.72	0.58	−1.8
$GG\{\beta\}$	0.39	0.82	0.72	−1.2
$\mathbb{C}(a, b, c)$	0.11	0.79	0.63	+2.3

Figure 8 depicts the cost function reduction based on LG function. As we can see, LG, based on M.D as constructed earlier in theory, decides the widening or shrinking of the green shape, as shown.

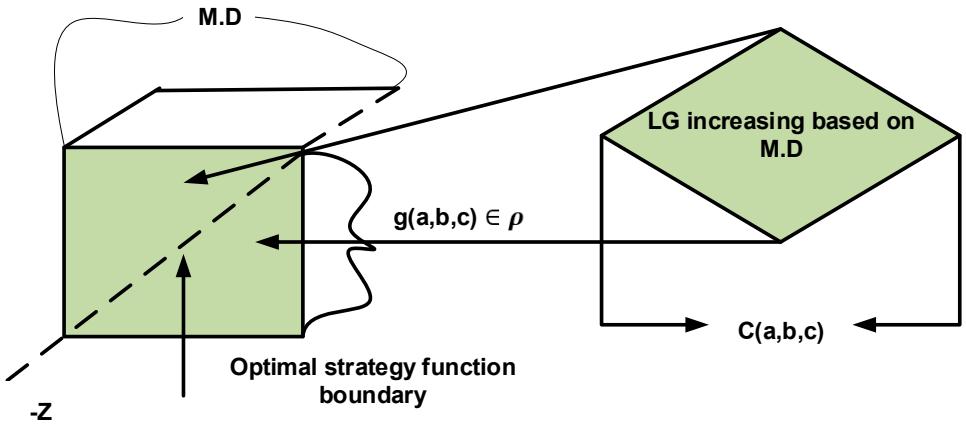


Figure 8. Cost function reduction using local gain for optimal strategy.

As the arrow moves in the 3-D space (cube) in either direction (optimum for z) then the COST function changes, for which the following conditions apply in optimum space in acceptable Gain function as explained earlier, such that $\mathbb{F} \in \underset{\mathbb{F} \in (\mathcal{R}+1)}{\min}(\Phi(t))$, for $t \to (T+1)^{\mathcal{R}}$, and $\mathcal{R}^n \to \mathcal{R}$ for infinite point in 3-D space.

Finally, we can write the generic form of our three constructs that this theorem aimed to develop. The following algorithm aims to govern these and provides to the external layer that can be built as:

$$lgo.Blend(x,y,z)) = \prod_{i=0}^{new\ N} \mathbb{F}_i + \sum_{j=0}^{new\ K} \mathbb{C}_j \left| \left(\mathbb{F}_i \times \mathbb{C}_j\right) \right| - \sum_{m,n,o}^{X,Y,Z} (A_m + A_n + A_0) \quad (30)$$

$$lgo.Blend(x,y,z)) = \prod_{i=0}^{new\ N} \mathbb{F}_i + \sum_{j=0}^{new\ K} \mathbb{C}_j \left| \left(\mathbb{F}_i \times \mathbb{C}_j\right) \right| - \sum_{m,n,o}^{X,Y,Z} (A_m + A_n + A_0) \quad (31)$$

$$\mathbb{E}((\mathbb{G}(x,y,z)) = Algo.Error(x,y,z)) = \sum_{i=1}^{g(x,y,z)} Algo.GG((\mathbb{G}(x,y,z)) - \sum_{j=1}^{err(x,y,z)} Err^2{}_j \quad (32)$$

The optimization of the model is critical. As a result, there is a requirement for experimental examination of various functions to be employed for improvement, as indicated in Table 5.

Table 5. Experimental analysis of the functions including GG function.

Metrics	1	2	3	4	5	6	7	8	9	10
$\mathbb{G}(GG)$	0.0112	0.0325	0.0321	0.0790	0.1390	0.2007	0.3892	0.5802	0.6419	0.6912
$\Omega(*,x\|y\|z)$	0.0012	0.0024	0.0079	0.0083	0.0097	0.0293	0.0301	0.0402	0.0511	0.5990
$g(x,y,z)$	0.2198	0.2500	0.2698	0.3312	0.3900	0.4319	0.4901	0.5510	0.5912	0.6339
O.F	0.9123	0.8121	0.8821	0.7102	0.65103	0.6615	0.5805	0.5210	0.5190	0.4231
U.F	0.7000	0.6812	0.6617	0.6101	0.5512	0.5100	0.4913	0.4301	0.4100	0.3809

Definition 1. *PTIP explains the principles of the two essential functions of tuning and blending. These constructions proved to be really helpful for classification objectives. Additionally, this definition builds the Binary decomposition function that the LT object uses to define and calculate the cost function.*

The Algorithm 3 builds on Theorem 3, which computes the complex error function, gain function, and Blend function. The Algorithm 3 lowers model parameter errors in order to produce a resilient framework. Figure 9 depicts the proposed PTIP at a higher elevation. The framework is an enhancement to the normal machine learning approach for training and assessing raw data. The innovative ideas of LE, GE, LG, and GG are presented in the final framework, which jointly aid in enhancement.

Algorithm 3: Blended and optimal computation for a resilient framework

Input: Φ, $\text{Risk}_{\text{emp}}[f]$, e, E, $\ddot{E}$, F{n}, The features set, Data Sets for Sampling
Output: $\mathbb{E}$, Algo.Blend(x, y, z)), Algo.GG((𝔾(x,y,z))
Initialization: $e \leftarrow 0$, $E \leftarrow 0\prime$

1. **Create** LTObject, $\text{NODES}_{i \in N}$, PTIP
2. **Generate** ObjectIMLE(h)/* Generate an object reference for improved machine learning engine application programming interface (API)*/
3. **Set** ObjectIMLE.PublicFunctions(h.IMLE, h.EFES, h.EWPM, h.ICCS)/* managing the four constructs*/
4. **Compute** Global SDE $\sigma \leftarrow \sqrt{\left(\frac{\sum_j \gamma^2{}_j}{(n-p-1)}\right)}$
5. **Initialize** Y_i, ϵ
6. **For** (each row in LTobject) **Do**
7. **Construct:** the $\mathbb{W}$ from the LT
8. **Construct** Logical 3D Triangle /* As shown in Figure 7 */
9. **Compute** LE /*Using equations (19) */
10. **While** (Distance Function Reduces and $\mathbb{E} \geq e$ && $E \in (0:1)$)) **Do**
11. **Compute** $E_d \leftarrow \sqrt{(x_1 - x_2)^2 + (y_1 - y_2)^2}$
12. **Compute** $H_d \leftarrow \sum_{j=1}^{k} \left| x_j - y_j \right|$
13. **Set** $P\{Y_i = y_i\} \leftarrow \pi_i^{y_i}(1 - \pi_i)^{1-y_i}$
14. **Update** LTObject (E_d, H_d)
15. **If** (E_d and $H_d < \sqrt{\mathbb{W}^{Y_i}}$) **then**
16. **Compute:** $M_d \leftarrow \left(\sum_{j=1}^{k} \left| x_j - y_j \right|^p\right)^{\frac{1}{p}}$
17. **Compute:** LG g(x, y, z)
18. **Compute** ObjectIMLE.Fitness(g.EFES)
19. **Recompute** E_d and H_d /* Based on g.EFES) */
20. **Set** Norm $\rightarrow (\alpha, \beta, \gamma)$
21. **Compute** 𝔾
22. **End If**
23. **Compute** $LF_1 \leftarrow \left(\frac{p_1}{1-p_1}\right)$
24. **Compute** $LF_2 \leftarrow \left(\frac{p_2}{1-p_2}\right)$
25. **Set** $\mathcal{R}^n \rightarrow \mathcal{R}$
26. **Set** $t \rightarrow (T+1)^{\mathcal{R}}$ /* T is averaged of t */
27. **Set** $e(L), e(H) \leftarrow (x \in X, y \in Y, z \in Z)$ /* Based on LT object values */
28. **End While**
29. **Examine** ObjectIMLE.Optimize(LF_1, LF_2, 𝔾
30. **Update** LTObject.Metrics(ObjectIMLE)
31. **Compute** $\mathbb{E}$, Algo.Blend(x, y, z)), Algo.GG((𝔾(x,y,z))
32. **End For**
33. **Return** $\mathbb{E}$, Algo.Blend(x, y, z)), Algo.GG((𝔾(x,y,z))

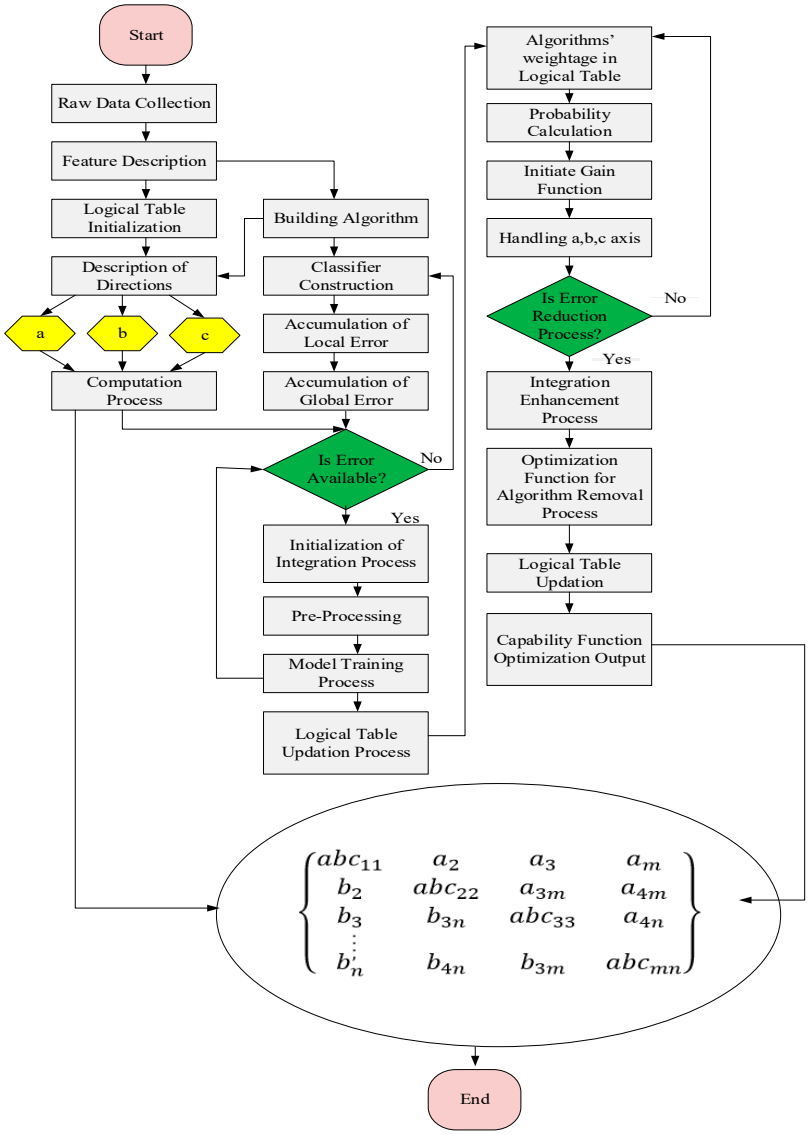


Figure 9. Illustration of elevated level for proposed PTIP.

Definition 2. *PTIP includes a thorough explanation of the Cost function theory based on Definition 1. This cost function is also essential for evaluating the algorithms, comparing components and features, as well as for computing the precise quantification of the blending and tuning functions.*

Case Study 1. The proposed PTIP solves two major issues, weaker generalization and reduced accuracy. The proposed PTIP can be expanded to popular machine learning scenarios. Generalization in machine learning shows how well a trained model categorizes or predicts previously unobserved data. When a machine learning model is generalized, it may be trained to function across all subsets of unknown data. One instance is the intricate clinical processes involved in medication manufacture. Particularly with today's high-tech treatments dependent on particular ingredients and manufacturing processes, medication creation takes time. The entire process is divided into several stages, some of which are contracted out to specialized suppliers. Therefore, these procedures need to be carried out accurately.

It has currently been observed with COVID-19 vaccine production that it requires generalization. The vaccine's creators provide the production blueprint, which is then carried out in sterile production facilities. The vaccine is transported in tanks to businesses that fill it in clinical settings in tiny dosages before another business produces the supply.

Additionally, medications may only be kept for a short period of time and frequently require particular storage conditions. Accurate treatment should be provided for these disorders. Clinical conditions can be accurately provided by the PTIP. The entire planning process is a very complex process that starts with having the appropriate input materials available at the appropriate time, continues with having enough production capacity, and ends with having the appropriate number of medications stored to meet demand. In addition, this has to be handled for countless treatments, each with a particular set of circumstances. The management of this complexity requires the use of computational approaches. For instance, PTIP with supervised learning techniques is used to choose the best partners for the production process.

4. Experimental Result and Discussion

This section presents various experimental results with necessary discussion and information. Each figure is accompanied with detailed information and comments to elaborate on the experimental analysis of the proposed model.

4.1. Research Methodology and Datasets

4.1.1. Research Methodology

Extensive research, acquaintance, and assessment are critical aspects in establishing the foundation of our proposed PTIP paradigm. As a result, we implemented the proposed PTIP and compared it to existing state-of-the-art algorithms, and as a further result, libraries used for existing algorithms are used for getting the outcomes. For diverse research topics and datasets we investigated the literature and reported their conclusions. Some of the results are compared to our proposed approach, as indicated in the results in the following sections. The goal of our study is to learn the current status of these ML algorithms so that we may progress them with our work, as detailed in this paper. To summarize, the genuine comparison is rather difficult due to the nature of the issue and datasets for which the algorithms are designed. In other words, one may outperform the other in some instances, but the reverse outcomes may be found in others. The scope of this article does not allow for a detailed comparison and experimental examination of each. A broad comparison of several of the algorithms is offered.

4.1.2. Datasets

We made use of information from the following several domains. Some datasets are available in raw, CSV, and SQL light formats, complete with parameters and field descriptions. All of our input data were converted and stored in the SQL Server database system. Some datasets have been discovered to be appropriate for performing tasks such as stock market, healthcare prevention, criminal control prediction, and epidemic identification.

- http://www.Kaggle.com, accessed on 1 August 2022. Iris species, credit card fraud detection, flight delays and cancellations, human resource analytics, daily news for market prediction, SMS spam collection, 1.88 Million US wildfires, gender classification, Twitter users, retail data Analytics, US mass shootings, breast cancer, exercise pattern prediction, fatal police shootings, Netflix prize data, student survey, Pima Indians diabetes Database, Zika virus epidemic, WUZZUF Job Posts.
- Social networking APIs.
- http://snap.standford.edu, accessed on 1 August 2022. Twitter, Facebook, bitcoin and Wiki dataset.
- https://aws.amazon.com/datasets/, accessed on 1 August 2022. Japan Census data, Enron email data, 1000 Genomics Project.
- http://archive.ics.uci.edu/ml/index.php, accessed on 1 August 2022. heart disease dataset, Iris, car evaluation, bank marketing data.
- https://docs.microsoft.com/en-us/azure/sql-database/sql-database-public-data-sets, accessed on 1 August 2022.

- https://www.reddit.com/r/bigquery/wiki/das, accessed on 1 August 2022.

4.1.3. Tools

The preliminary work is done using a Microsoft SQL Server and machine learning, and Microsoft Azure. The R language, C#, and Python were used to assess the efficacy of the proposed approach and underlying algorithms. Python libraries such as Pandas, SciPy, Matplotlib, Numpy, statsmodels, scikit-learn, fuel, SKdata, ScientificPython, and fuel, mILK have been implemented. Additionally, R libraries such as KlaR, gbm, RWeka, tree, CORELearn, ipred, rpart, MICE Package, CARET, PARTY, and random forest were utilized. Finally, GraphPad Prism is used to generate the simulated findings.

Based on the testing, we obtained the following interesting metrics:

- Accuracy with fitness factor;
- Relative Accuracy;
- PTIP with Fitness function;
- Error Reduction;
- Generalization measurement.

4.2. Accuracy

We ran 10 experiments (using 200 data samples) for various datasets to observe the overall performance of the algorithms: Support vector machine (SVMs), K-Nearest Neighbor (KNN), Logistic Regression, Multiple Linear Regression (MLR) [29], Decision tree (DR) [30], Graph-based boosting (GB) [31], and Random forest (RF) algorithm [32]. The PTIP is implemented and compared with existing techniques. The results are produced as depicted in Figure 10a. We used CART and ID3 for DT, and it has been observed that the proposed PTIP is found to be efficient. When the number of data samples is increased to more than 300+, the performance of the proposed PTIP still remains stable as compared to competing algorithms depicted in Figure 10b. On the other hand, the accuracy of the contending algorithms is greatly affected. It is imperative to note that we observed through our experiments that the outcome of various algorithms is not highly consistent. Hence, it is our motivation to pursue this research to stabilize the outcome and create a modified version that may be generalizable for many different problems and datasets.

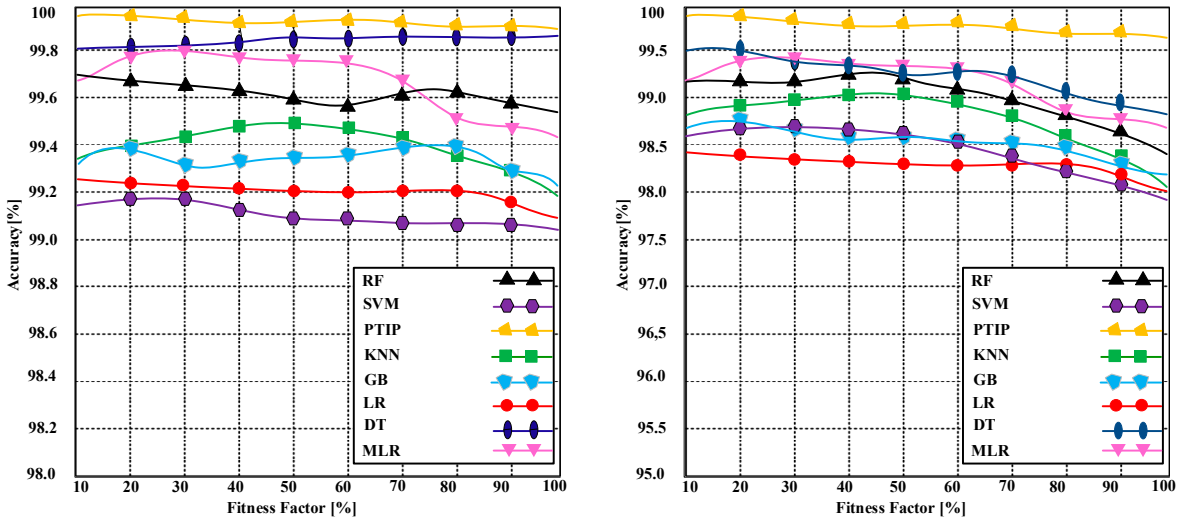


Figure 10. (**a**) Comparison of PTIP with existing algorithms based on correlation accuracy with fitness factor with 200 data samples; and (**b**) Comparison of PTIP with existing algorithms based on correlation accuracy with fitness factor with 300+ data samples.

4.3. Relative Accuracy

The relative accuracy shows that the measured value should be almost comparable to the standard value. The proposed method PTIP delivers the optimum fitting. Figure 11a shows how to calculate relative accuracy using error limits reflection. In the case of underfitting, outfitting, and global assessment error, a modest global gain is employed. The proposed PTIP's performance was compared to the following state-of-the-art methods: an iterative method for discrete variational (IMDV) [20]; cross-modal Turing test (CMTT) [21]; polynomial-time Turing machine (PTTM) [22]; neural Turing machine (NTM) [23]; and a-decidable and i-decidable algorithms (AIA) [24]. According to the experimental data, the PTIP generates the least relative accuracy with error bounds reflection.

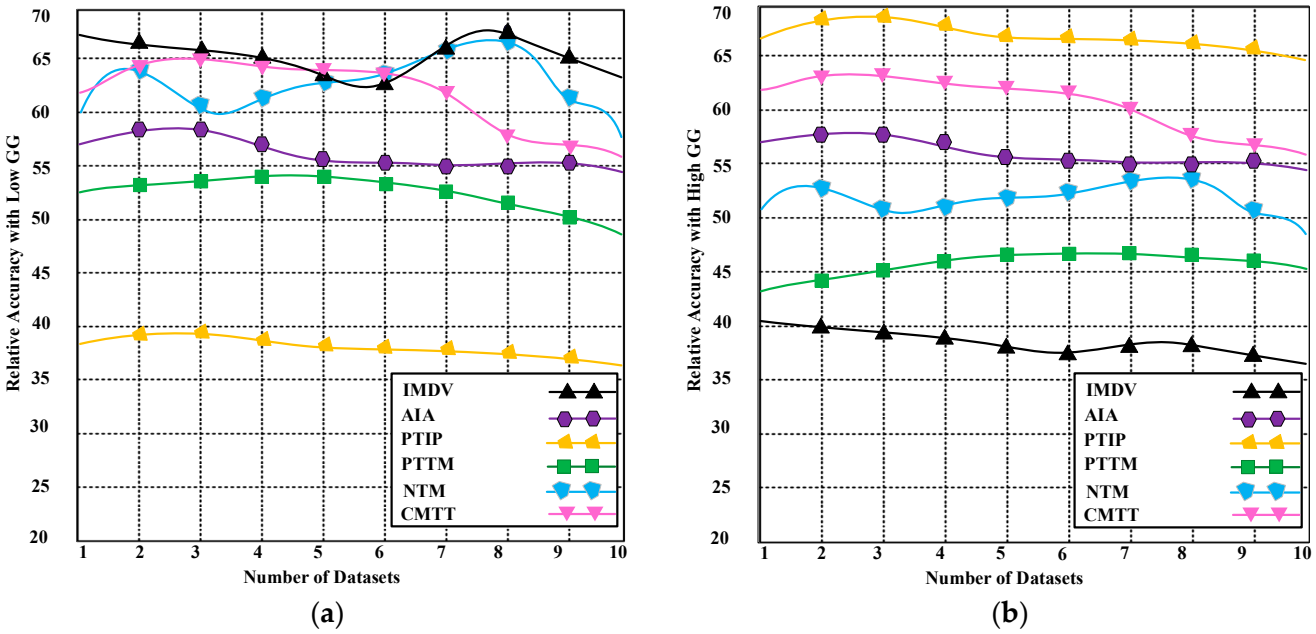


Figure 11. (**a**) demonstrates relative accuracy with low GG; and (**b**) demonstrates relative accuracy with high GG.

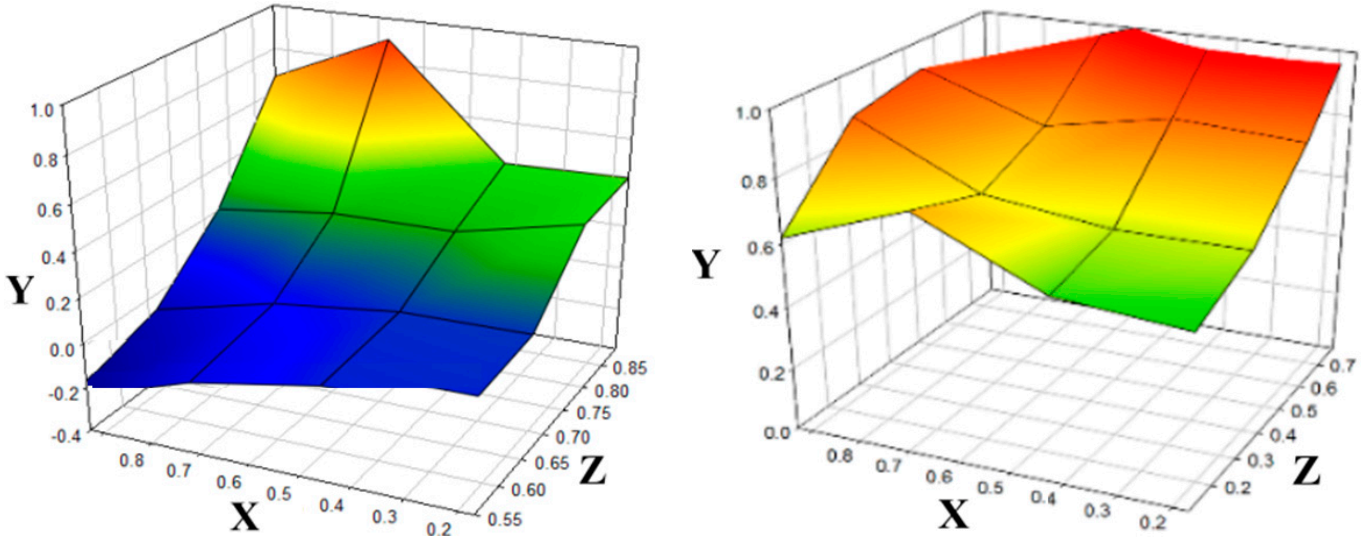
The lower relative accuracy with low GG is considered good. The PTIP obtains 38.76% relative accuracy with error bounds reflection. While the contending methods produce 48.7–61.2% relative accuracy with error bounds reflection, that is considered worse. Figure 11b demonstrates the relative accuracy with a high GG. Based on the result, it is observed that PTIP produces better relative accuracy that helps improve the classifier. The main reason for getting better relative accuracy with high GG is to use of improved optimum fitness function. The PTIP obtains 70.5% relative accuracy with high GG, which is considered an acceptable accuracy. On the other hand, the contending methods gain low relative accuracy with high GG which is considered worse. The contending methods obtain 42.4–55.3% relative accuracy with high GG. The findings were derived from 10 separate datasets using an experiment with an average of 20 runs.

4.4. Fitness Process

A fitness function is a type of objective function generated by the proposed PTIP in order to define a single figure of merit. The testing is carried out, and the findings are classified into four colors: in each dimension the blue signifies extreme overfitting.

Green and yellow show that the classifier is unable to distinguish between overfitting and underfitting, while orange shows the best fit. Figure 12a shows the best fitness capability in each dimension. The inaccuracy at the negative value has been noted based on the results. The perfect performance of the LT optimum fitness function is depicted in Figure 12b. The absence of the blue tint is noticeable. Furthermore, it indicates that the

z-dimension has the largest convergence, which is preferably required. As a consequence, the data are filtered using PTIP to find the optimal fitness range.



(**a**) (**b**)

Figure 12. (**a**) Illustration of a higher diffusion for entire fitness functions, which demonstrates extreme overfitting; (**b**) shows perfect performance of the LT optimum fitness.

4.5. Error Reduction

The PTIP is capable of reducing errors. The error reduction method is depicted in Figure 13a–d. Figure 13a depicts the gain function dispersed across ten experimental runs. This result shows that the blended function in PTIP is distributed randomly along all axes. It demonstrates that GG is exceedingly low, and the model's fitness is quite bad. Figure 13b depicts the progress of the PTIP as the number of errors decreases. Figure 13c depicts the tuning process's evolution. As can be observed, the x and y dimensions have faded, and z has outrun the fitness limitations. However, the yellow color in the z-axis indicates an inaccuracy of more than 80%, which is still unacceptable for model application. Figure 13d finally shows the error in bounds of the (20–80%) rule, and the z-dimension has finally been optimized. The values displayed for 1.0 are optimistic, and the realistic values are observed from ranging 0.6–0.85.

4.6. Generalization Measurement

The generalization measurement process is depicted in Figure 14a–d. The experiment for 200 random datasets has been demonstrated using the proposed PTIP for generalization improvement. The proposed PTIP's GE and LE functions are depicted. A circle represents the GE function, whereas a square represents the LE function. It should be highlighted that these findings come after measuring the model's accuracy as an ideal fit across many thousand iterations.

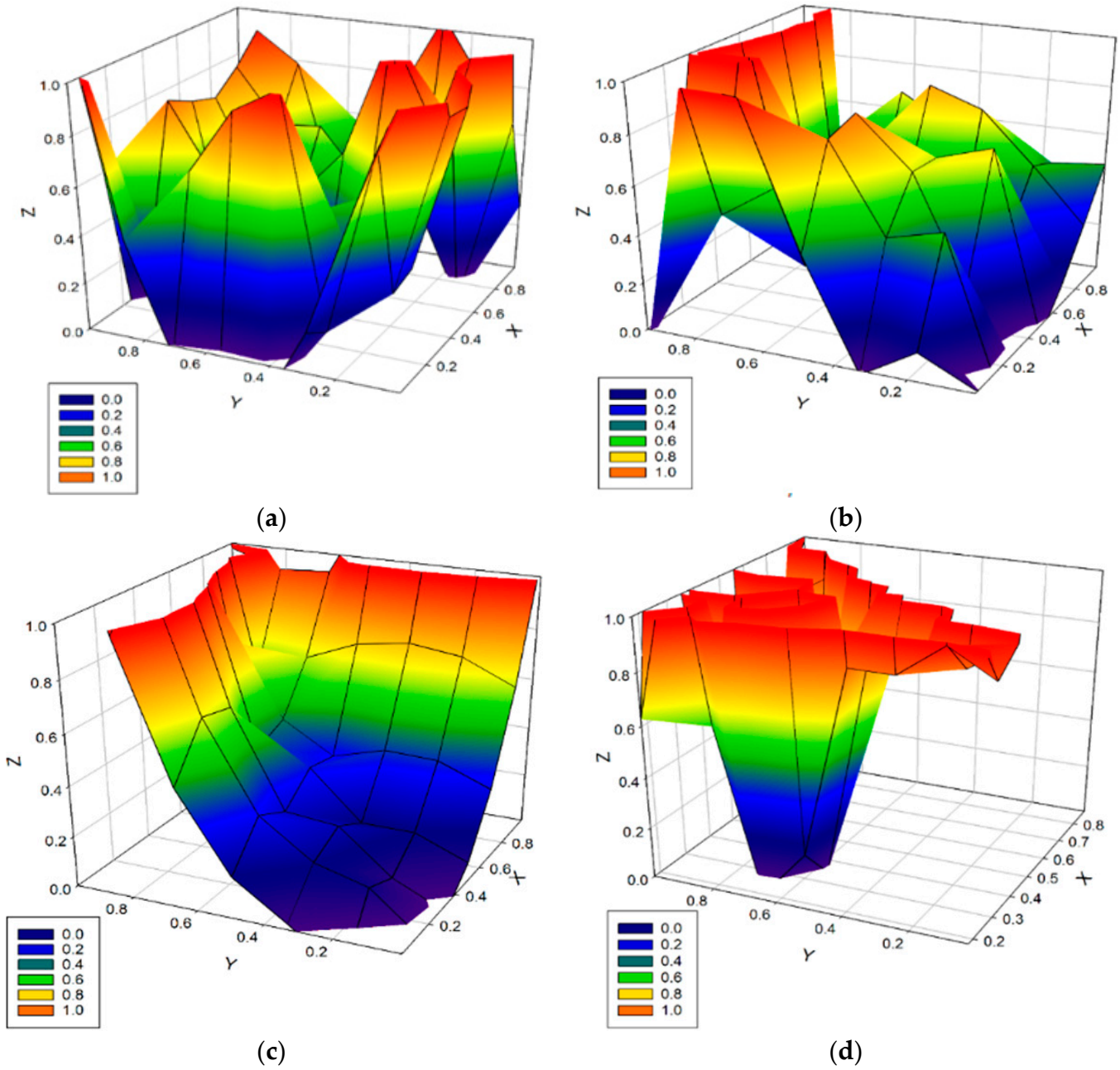


Figure 13. (**a**) shows spread of gain function; (**b**) shows the progress of the PTIP as the number of errors decreases; (**c**) displays the evolution of turning process; and (**d**) shows the error optimization process.

Figure 14a depicts the development of a random distribution of local and global complex error functions. Figure 14b depicts the better separation of the two functions. Figure 14c depicts the model gradually learning (self-teaching) the distinction of LE and GE. Finally, Figure 14d shows the optimal separation of both functions. It should be observed that both functions overlap since one is local and the other is global. It is concluded that the overall generalization is greatly improved.

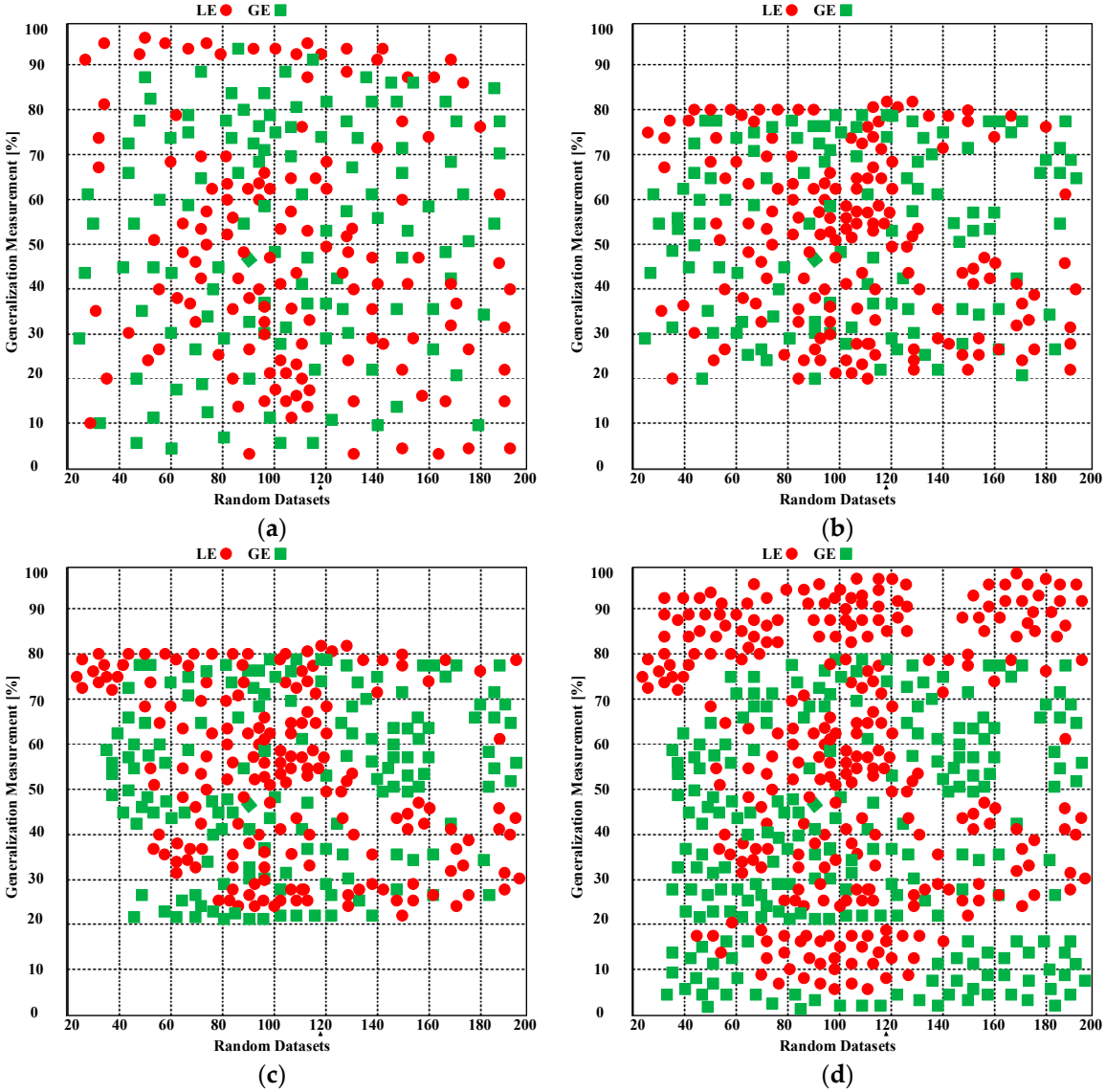


Figure 14. (a) demonstrates the generalization process for the random distribution of local and global complex error functions; (**b**) demonstrates the enhanced separation of the two functions (LE and GE); (**c**) demonstrates the self-learning process for the separation of LE and GE; (**d**) demonstrates the optimal separation of both functions and overlapping of LE and GE.

5. Conclusions and Future Work

This section summarizes the whole outcome of the proposed approach as well as its benefits.

5.1. Conclusions

The parallel Turing integration paradigm is described in this study to facilitate dataset storage. The proposed PTIP approach comprises of three cutting-edge building blocks: (a) it demonstrates the expansion of a logical 3-D cube that administers the algorithms to ensure optimal performance; (b) it demonstrates the improvement in a blend of algorithms based on parallel tuning of model and classifiers; and (c) it provides the ultimate model engineering to learn from its errors (wrong predictions) and explain itself to choose the correct algorithm and eliminate the incorrect predictions.

The outcomes of several tests are discussed in this article. To fine-tune the model and produce simulated data for the study, we ran 10–20 trials. In 3D space, the LG and GG

functions have been developed and improved. Finally, the proposed PTIP offered better outcomes with the ability to generalize on a particular collection of facts and issues.

5.2. Future Work

We will test more algorithms, particularly in the fields of unsupervised learning, to enhance the PTIP and its components. We will be improving/developing a model called the "Predicting Educational Relevance For an Efficient Classification of Talent (PERFECT) algorithm Engine" (PAE). The PAE may be used in conjunction with PTIP and includes three cutting-edge algorithms: Good Fit Student (GFS); Noise Removal and Structured Data Detection (NR-SDD); and Good Fit Job Candidate (GFC).

Author Contributions: A.R.: conceptualization, writing, idea proposal, software development, methodology, review, manuscript preparation, visualization, results and submission; M.B.H.F. and M.A. (Mohsin Ali): data curation, software development, and preparation; F.A. and N.Z.J.: review, manuscript preparation, and visualization; G.B., M.A. (Muder Almi'ani), A.A., M.A. (Majid Alshammari) and S.A.: review, data curation and editing. All authors have read and agreed to the published version of the manuscript.

Funding: This work was supported by Taif University Researchers Supporting Project number (TURSP-2020/302), Taif University, Taif, Saudi Arabia.

Data Availability Statement: The data that support the findings of this research are publicly available as indicated in the reference.

Acknowledgments: The authors gratefully acknowledge the support of Mohsin Ali for providing insights.

Conflicts of Interest: The authors declare no conflict of interest.

References

1. Duan, B.; Yuan, J.; Yu, C.-H.; Huang, J.; Hsieh, C.-Y. A survey on HHL algorithm: From theory to application in quantum machine learning. *Phys. Lett. A* **2020**, *384*, 126595. [CrossRef]
2. Ho, S.Y.; Wong, L.; Goh, W.W.B. Avoid oversimplifications in machine learning: Going beyond the class-prediction accuracy. *Patterns* **2020**, *1*, 100025. [CrossRef] [PubMed]
3. Razaque, A.; Amsaad, F.; Halder, D.; Baza, M.; Aboshgifa, A.; Bhatia, S. Analysis of sentimental behaviour over social data using machine learning algorithms. In Proceedings of the Advances and Trends in Artificial Intelligence. Artificial Intelligence Practices: 34th International Conference on Industrial, Engineering and Other Applications of Applied Intelligent Systems, IEA/AIE 2021, Kuala Lumpur, Malaysia, 26–29 July 2021; Springer International Publishing: Berlin/Heidelberg, Germany; pp. 396–412.
4. Uddin, M.F.; Rizvi, S.; Razaque, A. Proposing logical table constructs for enhanced machine learning process. *IEEE Access* **2018**, *6*, 47751–47769. [CrossRef]
5. Razaque, A.; Ben Haj Frej, M.; Almi'ani, M.; Alotaibi, M.; Alotaibi, B. Improved support vector machine enabled radial basis function and linear variants for remote sensing image classification. *Sensors* **2021**, *21*, 4431. [CrossRef]
6. Yang, Y.; Wu, L. Machine learning approaches to the unit commitment problem: Current trends, emerging challenges, and new strategies. *Electr. J.* **2021**, *34*, 106889. [CrossRef]
7. Kiselev, P.; Kiselev, B.; Matsuta, V.; Feshchenko, A.; Bogdanovskaya, I.; Kosheleva, A. Career guidance based on machine learning: Social networks in professional identity construction. *Procedia Comput. Sci.* **2020**, *169*, 158–163. [CrossRef]
8. Zhao, L.; Ciallella, H.L.; Aleksunes, L.M.; Zhu, H. Advancing computer-aided drug discovery (CADD) by big data and data-driven machine learning modeling. *Drug Discov. Today* **2020**, *25*, 1624–1638. [CrossRef]
9. Belmonte, L.; Segura-Robles, A.; Moreno-Guerrero, A.-J.; Parra-González, M.E. Machine learning and big data in the impact literature. A bibliometric review with scientific mapping in Web of science. *Symmetry* **2020**, *12*, 495. [CrossRef]
10. Jiang, J. Selection Bias in the Predictive Analytics with Machine-Learning Algorithm. *Ann. Emerg. Med.* **2021**, *77*, 272–273. [CrossRef]
11. Lure, A.C.; Du, X.; Black, E.W.; Irons, R.; Lemas, D.J.; Taylor, J.A.; Lavilla, O.; de la Cruz, D.; Neu, J. Using machine learning analysis to assist in differentiating between necrotizing enterocolitis and spontaneous intestinal perforation: A novel predictive analytic tool. *J. Pediatr. Surg.* **2020**, *56*, 1703–1710. [CrossRef]
12. Xiong, D.; Fu, W.; Wang, K.; Fang, P.; Chen, T.; Zou, F. A blended approach incorporating TVFEMD, PSR, NNCT-based multi-model fusion and hierarchy-based merged optimization algorithm for multi-step wind speed prediction. *Energy Convers. Manag.* **2021**, *230*, 113680. [CrossRef]
13. Lin, S.; Kim, J.; Hua, C.; Park, M.-H.; Kang, S. Coagulant dosage determination using deep learning-based graph attention multivariate time series forecasting model. *Water Res.* **2023**, *232*, 119665. [CrossRef]

14. Schorlemmer, M.; Plaza, E. A uniform model of computational conceptual blending. *Cogn. Syst. Res.* **2021**, *65*, 118–137. [CrossRef]
15. Messaoud, S.; Bradai, A.; Bukhari, S.H.R.; Qung, P.T.A.; Ahmed, O.B.; Atri, M. A Survey on Machine Learning in Internet of Things: Algorithms, Strategies, and Applications. *Internet Things* **2020**, *12*, 100314. [CrossRef]
16. Basha, S.M.; Rajput, D.S. Survey on Evaluating the Performance of Machine Learning Algorithms: Past Contributions and Future Roadmap. In *Deep Learning and Parallel Computing Environment for Bioengineering Systems*; Academic Press: Cambridge, MA, USA, 2019; pp. 153–164.
17. Gambella, C.; Ghaddar, B.; Naoum-Sawaya, J. Optimization problems for machine learning: A survey. *Eur. J. Oper. Res.* **2020**, *90*, 807–828. [CrossRef]
18. Simsekler, M.E.; Rodrigues, C.; Qazi, A.; Ellahham, S.; Ozonoff, A. A comparative study of patient and staff safety evaluation using tree-based machine learning algorithms. *Reliab. Eng. Syst. Saf.* **2021**, *208*, 107416. [CrossRef]
19. Tandon, N.; Tandon, R. Using machine learning to explain the heterogeneity of schizophrenia. *Realiz. Promise Avoid. Hype. Schizophr. Res.* **2019**, *214*, 70–75.
20. Ferraro, S.J.; de Diego, D.M.; Martín de Almagro, R.T.S. Parallel iterative methods for variational integration applied to navigation problems. *IFAC-Pap.* **2021**, *54*, 321–326. [CrossRef]
21. Leshchev, S.V. Cross-modal Turing test and embodied cognition: Agency, computing. *Procedia Comput. Sci.* **2021**, *190*, 527–531. [CrossRef]
22. Pinon, B.; Jungers, R.; Delvenne, J.-C. PAC-learning gains of Turing machines over circuits and neural networks. *Phys. D Nonlinear Phenom.* **2023**, *444*, 133585. [CrossRef]
23. Faradonbe, S.M.; Safi-Esfahani, F. A classifier task based on Neural Turing Machine and particle swarm algorithm. *Neurocomputing* **2020**, *396*, 133–152. [CrossRef]
24. Eberbach, E. Undecidability and Complexity for Super-Turing Models of Computation. *Proceedings* **2022**, *81*, 123.
25. Kashyap, K.; Siddiqi, M.I. Recent trends in artificial intelligence-driven identification and development of anti-neurodegenerative therapeutic agents. *Mol. Divers.* **2021**, *25*, 1517–1539. [CrossRef]
26. Mohammadi, M.; Rashid, T.A.; Karim, S.H.T.; Aldalwie, A.H.M.; Tho, Q.T.; Bidaki, M.; Rahmani, A.M.; Hoseinzadeh, M. A comprehensive survey and taxonomy of the SVM-based intrusion detection systems. *J. Netw. Comput. Appl.* **2021**, *178*, 102983. [CrossRef]
27. Hu, J.; Peng, H.; Wang, J.; Yu, W. kNN-P: A kNN classifier optimized by P systems. *Theor. Comput. Sci.* **2020**, *817*, 55–65. [CrossRef]
28. Nabi, M.M.; Shah, M.A. A Fuzzy Approach to Trust Management in Fog Computing. In Proceedings of the IEEE 2022 24th International Multitopic Conference (INMIC), Islamabad, Pakistan, 21–22 October 2022; pp. 1–6.
29. Tang, W.; Li, Y.; Yu, Y.; Wang, Z.; Xu, T.; Chen, J.; Lin, J.; Li, X. Development of models predicting biodegradation rate rating with multiple linear regression and support vector machine algorithms. *Chemosphere* **2020**, *253*, 126666. [CrossRef]
30. Zhou, H.F.; Zhang, J.; Zhou, Y.; Guo, X.; Ma, Y. A feature selection algorithm of decision tree based on feature weight. *Expert Syst. Appl.* **2021**, *164*, 113842. [CrossRef]
31. Liu, Z.; Jin, W.; Mu, Y. Graph-based boosting algorithm to learn labeled and unlabeled data. *Pattern Recognit.* **2020**, *106*, 107417. [CrossRef]
32. Jain, N.; Jana, P.K. LRF: A logically randomized forest algorithm for classification and regression problems. *Expert Syst. Appl.* **2023**, *213*, 119225. [CrossRef]