ENHANCED MACHINE LEARNING ENGINE ENGINEERING USING INNOVATIVE BLENDING, TUNING, AND FEATURE OPTIMIZATION

Muhammad Fahim Uddin

Under the Supervision of Dr. Jeongkyu Lee

DISSERTATION

SUBMITTED IN PARTIAL FULFILMENT OF THE REQUIREMENTS

FOR THE DEGREE OF DOCTOR OF PHILOSOPHY IN COMPUTER SCIENCE

AND ENGINEERING

THE SCHOOL OF ENGINEERING

UNIVERSITY OF BRIDGEPORT

CONNECTICUT

Nov, 2018

ENHANCED MACHINE LEARNING ENGINE ENGINEERING USING INNOVATIVE BLENDING, TUNING, AND FEATURE OPTIMIZATION

Muhammad Fahim Uddin

Under the Supervision of Dr. Jeongkyu Lee

Approvals

Committee Members

Name	Signature	Date
Dr. Jeongkyu Lee		1/27/2019
Dr. Navarun Gupta		2/4/19
Dr. Xingguo Xiong		02/05/2019
Dr. Abdelshakour Abuzneid		2/4/2019
Dr. Syed Rizvi		1/28/2019

Ph.D. Program Coordinator

Dr. Khaled M. Elleithy		2/5/19

Chairman, Computer Science and Engineering Department

Dr. Ausif Mahmood		2/5/2019

Dean, School of Engineering

Dr. Tarek M. Sobh		2/5/2019

ENHANCED MACHINE LEARNING ENGINE ENGINEERING USING INNOVATIVE BLENDING, TUNING, AND FEATURE OPTIMIZATION

ENHANCED MACHINE LEARNING ENGINE ENGINEERING USING INNOVATIVE BLENDING, TUNING, AND FEATURE OPTIMIZATION

ABSTRACT

Investigated into and motivated by Ensemble Machine Learning (*ML*) techniques, this thesis contributes to addressing performance, consistency, and integrity issues such as overfitting, underfitting, predictive errors, accuracy paradox, and poor generalization for the *ML* models. Ensemble *ML* methods have shown promising outcome when a single algorithm failed to approximate the true prediction function. Using meta-learning, a super learner is engineered by combining weak learners. Generally, several methods in Supervised Learning (*SL)* are evaluated to find the best fit to the underlying data and predictive analytics (i.e., "*No Free Lunch"* Theorem relevance). This thesis addresses three main challenges/problems, *i)* determining the optimum blend of algorithms/methods for enhanced *SL* ensemble models, *ii)* engineering the selection and grouping of features that aggregate to the highest possible predictive and non-redundant value in the training data set, and *iii)* addressing the performance integrity issues such as accuracy paradox. Therefore, an enhanced Machine Learning Engine Engineering (*eMLEE)* is inimitably constructed via built-in parallel processing and specially designed novel constructs for

error and gain functions to optimally score the classifier elements for improved training experience and validation procedures. *eMLEE,* as based on stochastic thinking, is built on; *i)* one centralized unit as Logical Table unit (*LT*), *ii)* two explicit units as enhanced Algorithm Blend and Tuning (*eABT*) and enhanced Feature Engineering and Selection (*eFES*), and two implicit constructs as enhanced Weighted Performance Metric(*eWPM*) and enhanced Cross Validation and Split *(eCVS*). Hence, it proposes an enhancement to the internals of the *SL* ensemble approaches.

Motivated by nature inspired metaheuristics algorithms (such as *GA, PSO, ACO*, etc.), feedback mechanisms are improved by introducing a specialized function as *Learning from the Mistakes* (*LFM*) to mimic the human learning experience. *LFM* has shown significant improvement towards refining the predictive accuracy on the testing data by utilizing the computational processing of wrong predictions to increase the weighting scoring of the weak classifiers and features. *LFM* further ensures the training layer experiences maximum mistakes (i.e., errors) for optimum tuning. With this designed in the engine, stochastic modeling/thinking is implicitly implemented.

Motivated by OOP paradigm in the high-level programming, *eMLEE* provides interface infrastructure using *LT* objects for the main units (i.e., Unit A and Unit B) to use the functions on demand during the classifier learning process. This approach also assists the utilization of *eMLEE* API by the outer real-world usage for predictive modeling to further customize the classifier learning process and tuning elements trade-off, subject to the data type and end model in goal.

Motivated by higher dimensional processing and Analysis (i.e., *3D*) for improved analytics and learning mechanics, *eMLEE* incorporates *3D* Modeling of fitness metrics

such as *x* for overfit, *y* for underfit, and *z* for optimum fit, and then creates logical cubes using *LT* handles to locate the optimum space during ensemble process. This approach ensures the fine tuning of ensemble learning process with improved accuracy metric.

To support the built and implementation of the proposed scheme, mathematical models (*i.e., Definitions, Lemmas, Rules, and Procedures*) along with the governing algorithms' definitions (*and pseudo-code*), and necessary illustrations (*to assist in elaborating the concepts*) are provided. Diverse sets of data are used to improve the generalization of the engine and tune the underlying constructs during development-testing phases. To show the practicality and stability of the proposed scheme, several results are presented with a comprehensive analysis of the outcomes for the metrics (i.e., *via integrity, corroboration, and quantification*) of the engine. Two approaches are followed to corroborate the engine, *i)* testing inner layers (i.e., internal constructs) of the engine (i.e., *Unit-A, Unit-B and C-Unit*) to stabilize and test the fundamentals, and *ii)* testing outer layer (i.e., *engine as a black box*) for standard measuring metrics for the real-world endorsement. Comparison with various existing techniques in the state of the art are also reported. In conclusion of the extensive literature review, research undertaken, investigative approach, engine construction and tuning, validation approach, experimental study, and results visualization, the *eMLEE* is found to be outperforming the existing techniques most of the time, in terms of the classifier learning, generalization, metrics trade-off, optimum-fitness, feature engineering, and validation.

DEDICATION

To

My Adored Wife

GUL MEENA

Who astronomically cooperated While PhD work continued

Machine Learning is more **art** than science.

Let us optimize this **art** through science.

ACKNOWLEDGEMENTS

Top of all, I thank **ALMIGHTY GOD**, who made me a part of His creation, thus providing me an opportunity to learn from the knowledge, He created. He enabled me to evolve into the rational being via the Intellect and the Reasons, He engineered around our consciousness.

Thanks to Dr. JEONGKYU LEE for his terrific advisory and guidelines to constructively drill through the research studies over the course of PhD investigative work, Research Problem Identification, Implementation of the Proposed Solution, Preparation of the thesis book, and the Oral exams.

Thanks to Dr. AUSIF MAHMOOD who initially motivated me to consider my programing skills, logical intuition, and teaching spirit for the research in the Data Science areas. His outstanding teaching practices and deliverance of the technical concepts momentously helped me to learn the useful and relevant concepts for the pursuance of the PhD research work.

Thanks to Dr. KHALED ELLEITHY for his formal management of the PhD program. He also consistently encouraged me to work harder and consider great journals for publishing the quality write-up. That further resulted in an enhancement of the PhD research work.

Thanks to the committee members; Dr. NAVARUN GUPTA, Dr. XINGGUO XIONG, Dr. ABDELSHAKOUR ABUZNEID, and Dr. SYED RIZVI for their continued advice, comments, time, and judgment for the oral exams.

Thanks to the contributors of data repositories online, without which, this research would not have evolved into the current state.

Thanks to reviewers of the journals and conferences papers for providing their constructive feedback and criticism that helped to improve the quality of the paper's write-up.

Thanks to my parents, relatives, and friends for their continuous support, wait, and the prayers.

Thanks to the University of Bridgeport, School of Engineering, that provided me with an admission to the PhD program.

KEY ACCRONYMS

Introduced

eMLEE	Enhanced Machine Learning Engine Engineering
eABT	Enhanced Algorithm Blending and Tuning
eFES	Enhanced Feature Engineering and Selection
eWPM	Enhanced Weighted Performance Metric
eCVS	Enhanced Cross Validation and Split
LT	Logical Table
FO	Feature Optimization
CF	Correlating Factor
FF	Fitness Factor
LFM	Learning From Mistakes
AF	Accepted Features
RF	Rejected Features
PF	Predicted Features
PV	Predictive Value
MF	Matching Factor
AF	Approximation Function
EFE	Extended Feature Engineering
Ov.F	Optimum Fitness
U.F	Under Fitness
O.F	Over Fitness
RoOpFit	Rule of Optimum Fitness
FWM	Feature Weights Mechanism
AP	Algorithm Pool
LE	Local Error
GE	Global Error
LG	Local Gain
GG	Global Gain

Standard

ML	Machine Learning
SL	Supervised Learning
DS	Data Science
DT	Decision Trees

SVM	Support Vector Machines
LReg	Linear Regression
MLR	Multiple Linear Regression
LR	Logistic Regression
KNN	K-Nearest Neighbors
RF	Random Forest
XGB	XGBoost
R-GBM	Random GBM
A-BOOST	Adaboost
IG	Information Gain
CS	Chi-squared
PC	Pearson Correlation
ANOVA	Analysis of Variance
WOE	Weight of Evidence
RFE	Recursive Feature Elimination
SFS	Sequential Feature Selector
US	Univariant Selection
PCA	Principal Component Analysis
RF	Random Forest
LASSO	Least Absolute Shrinkage and Selection Operator
RR	RIDGE Regression
EN	Elastic Net
GBM	Gradient Boosted Machines
LDA	Linear discriminant analysis
NNMF	Non-negative matrix factorization
JMI	Joint Mutual Information
ELM	Extreme Learning Machines
AL	Active Learning
MCSs	Multiple Classifier Systems
OELM	Optimization Extreme Learning Machine
SFM	Sample Feature Machine

KEY NOTATIONS

A_n	nth method
x,y,z	Point of overfitting (OF), underfitting(UF), Optimum-fitting(Op.F)
F_n	Complete raw feature set
F_z	Optimized feature set
υ_z	Optimized participating algorithm function
$\mathfrak{C}$	2D array
w	Unit weighted parameters to hold values in the array
O^m	Array optimization function for ^{m}th index
$\mathcal{L}$	Training parameter
ϕ	Blending Function
β	Optimum-fitness function factor
BIN	Binary classification function
(al)	Test algorithm
Al	Absolute value of the test algorithm fitness score
∇d	Euclidean distance function
$\mathbb{S}$	Suitability Score
μ	Distance unit
$\mathbb{s}$	Signal (useful data element)
$\mathbb{m}$	Noise (non-useful data element)
Ψ	Loss regulating factor in LT
$QUAN$	A quantizing function for factor determination
$\mathbb{B}$	Best scoring state of a test algorithm
μ	Probability function for a test
err(e), LE	Local error
Err(e), GE	Global error
g *(LG)*	Local gain
$\mathbb{G}$*(GG)*	Global gain
$\psi(n)$	Classifier function
$\|\|M\|\|(x,y,z)$	Norm function
Op.F	Optimum Factor
UF, OF	Underfit, Overfit Pointer
$\mathcal{R}$	Minimum Risk Function
$MF(x,y,z)$	Matching function based on each dimension
$\Delta_{(x,y,z)}^{(a,b)}$	Axis alignment method

$\mathbb{W}$	eMLEE based performance metric (eWPM)
σ	Standard deviation error
$\mathbb{H}, \mathbb{Z}$	LFM (learn from mistakes) module internal functions
x	Point of Overfitting(OF)
y	Point of underfitting (UF)
z	Point of optimum-fitting (OpF)
F_n	Complete raw feature set
$+\mathbb{F}$	Feature Adder Function
$-\mathbb{F}$	Feature Remover Function
LT	Logical Table Function
$A\ (i)$	ith ML algorithm such as SVM
$\mathbb{R}_{eFES}$	Ratio of normalized error between local and global errors
$F_{ran}\ (x, y, z)$	Randomized feature set
f_w	Weighted feature value
$\Delta(x, y, z)$	Regulating Function in LT object to obey the reference of 50% for training
err (e), LE	Local error
Err (e), GE	Global error
$\emptyset$	maximum inconsistency
Q^N	nth random generator
f_{ii}	Position of a feature in 2D space
g *(LG)*	Local gain
$\mathbb{G}$ *(GG)*	Global gain
af_i	ith accepted feature
rf_i	ith rejected feature
pf_i	ith predictive feature
ΔS_i	ith dataset item
$\nabla\ (\varphi, \rho, \omega)$	Acceptable parameter function for x, y, z
ObF	Objective function
$k \in K$	Predictor ID in the group of K
EC	Evaluation Criterion
$W(\emptyset)$	Weighted Function
N_γ, S_γ	Border unit normal vectors
$\widehat{\mathbb{Q}_\gamma}\ (\Delta)$	Probability distribution based on nonparametric density estimation
$Gain_I(w)$	Information gain
$J_{MIN}(Z)^d$	Jacobian minimization

LIST OF INTERNAL ALGORITHMS

LIST OF DEFINITIONS, LEMMAS, PROCEDURES, AND RULES

DEFINITIONS

LEMMAS

PROCEDURES

RULES

SELECTED STANDARD MEASURES FOR ENGINE CORROBORATION

	MEASURE	DEFINITION
1	Accuracy	Quality of being correct
2	Precision	Expresses the proportion of the data points the model says was relevant were relevant.
3	Recall	The ability of a model to find all the relevant cases within a dataset
4	Sensitivity	The probability (a posteriori) of its yielding true-positive (TP) results
5	Specificity	The probability (a posteriori) of its yielding negative results
6	Averaged Error	State of being wrong
7	F-measure	To measure the test's accuracy
8	Gini Coefficient	Used in collection of prediction for segregation
9	Correlation Coefficient	To determine the strength of relationship between elements or variables
10	AUC	To determine which of the used models predicts the classes best

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

CHAPTER 1 – INTRODUCTION

A tremendous growth in the digital data and needs for deeper insights, extracting new knowledge, and accurate predictions prompt us to consider the following general question:

"How well can we engineer the internals of the ensemble approaches in machine learning for optimized feature set, improved reliability of the accuracy, and model generalization?"

1.1 Background

Machine learning *(ML)* unveils tremendous potential in the data science and predictive analytics. *ML* algorithms[1][2] specially in supervised learning (*SL*) zones have advanced into improved modeling of the underlying data for decision making[3], predictive analytics[4], and personality prediction[5] etc. Some of the surveys such as [6][7][8] including domains of unstructured data[9] and social networking[10][11] platforms have shown the incredible importance of *ML* algorithms' improvements[12].

ML models is highly dependent on the underlying algorithms and their performances[13]. Overfitting, underfitting, low accuracy, poor generalization, low reliability, and predictive errors are some of the challenges in *ML*[14][15][16]. Overfitting perhaps is the toughest challenge being researched for a long time[17]. Despite several

algorithms/models developed over many decades, models/algorithms do not always fit the real-world problems and data[14]. This often results in the challenges stated above.

One of the most important research directions of *ML* is Feature Optimization (*FO*) (collectively grouped as Feature Engineering (*FE*), Feature Selection (*FS*), and Filtering)[18]. For *FS*, a saying *"Less is More*" becomes the essence of such research. Dimensionality Reduction [19] has become a focus in the *ML* process to avoid unnecessary computing power/cost, overlearning, and predictive errors. Redundant features which may have similar predictive value to other feature(s), may be excluded without negatively affecting the learning process and the irrelevant features should be excluded [20][21]. *FS* and *FE* not only focus on extracting a subset from the optimal feature set but also build new feature sets previously overlooked by *ML* techniques[22][23]. This also includes reducing the higher dimensions into lower ones to extract the feature's value[24]. Latest research has shown noteworthy progress in *FE*. In [25], the authors reviewed the latest progress in *FS* and associated algorithms. Out of a few, principal component analysis (PCA)[26] , Bayes error probability[27], Discriminant analysis, and Karhunen Loeve expansion[28] are widely used.

Evaluation, validation, integrity check, and performance measurements bring challenges of their own[29][30]. Models need to be thoroughly tested and validated with testing data to ensure reliability of the outcome, integrity of the accuracy measure and predictive errors. Accuracy paradox is the main challenge in such measurements. Significant research is being undertaken to improve such measures tradeoff and improve the reliability of the performance metrics[31][32][33].

1.2 Motivation

On broader level, our research was motivated by the following:

i) ***Nature inspired meta-heuristics algorithms*** – Similar to Genetic Algorithms, Particle Swarm Intelligence, Ant Colony Optimization, Evolutionary algorithms, Simulated Annealing, Greedy Algorithms, and Artificial Immune Systems, we adapted human learning from the mistake experience idea into our research.

ii) ***OOP Paradigm*** – We constructed our engine around the ideology of OOP model. The fundamentals of OOP motivated our engine to be adaptable, extensible, customizable, applicable, and re-useable.

iii) ***Higher Dimensional Analysis*** – Contrary to focusing on *1D or 2D* of measurement, such as accuracy or traditional table view, we adopted the idea of *3D* (higher dimensional analysis) of measuring the internal learning performance by quantifying the accuracy in *3D* Coordinates (*x:overfit, y:underfit, and z:optimum fit*) to increase the integrity of the accuracy measure for the model to generalize better when put to test.

iv) ***Ensemble Approaches*** – Based on "collective wisdom" and "unity is strength" philosophy, we were motivated by the power of boosting and bagging techniques.

v) ***Stochastic Thinking and Math*** – Since *ML* brings lot of uncertainty in the model's behavior. Like the saying goes "All models are wrong in *ML*, but some are useful". Math can't predict what it has not seen. Therefore, we were motivated to investigate the aggregation of mathematical tricks and tools into stochastic modeling of *ML*.

The early stages of our research was also motivated by the latest research in personality computing[6] with focus on talent recognition from social networking data in conjunction with the academic and career data to advance the predictive analytics and validating the recommendation system models. To envision our proposed scheme, we investigated the concepts and progress of various *SL* methods[34][35][36][37][38][39][40]. The latest progress in our research was further motivated by ensemble approaches[41]. We further investigated some of the recent developments in ensemble methods with focus on boosting[42](pioneer of Adaboost), bagging techniques and related outstanding problems[43][44][45][46]. Our comprehensive literature review, briefly presented in *Chapter 2*, has revealed the potential of these existing algorithms to enhance the predictive analytics and classifier learning particularly with ensemble techniques. Despite using the best models and algorithms, *FO* is crucial to the performance of the *ML* process and predictions. *FS* has been a focus in the fields of data mining[47], data discovery, text classification[48], and image processing[49]. Unfortunately, raw datasets pose no clear advice or insight into which variables must be focused on. Usually, datasets contain several variables/features but not all of them contribute towards predictive modeling. Relevant feature selection[50][51], feature extraction[52], feature relevancy[23], and feature

engineering has been a focus of the recent research. Another significance of such research is to determine the intra- and inter-relationships between the features. Their internal dependence and correlation/relevance greatly impact the way a model learns from the data[16]. To make the process computationally inexpensive and keep the accuracy higher, features should be categorized by the algorithm itself. The existing literature proves that such work is rarely undertaken in *ML* research.

1.3 Research Questions

In compliance with the latest notable work[53][54][55][14], the following general research questions enlighten the directions for our research work and formulating our research problems and hypothesis statement.

1. Can error rules be created for upper and lower bounds to govern the classifier learning in optimum fitting range?
2. Can tuning be enhanced in parallel-mode governed by measures of the model?
3. Can a model (i.e., blended) learn from its mistakes (i.e., wrong predictions)
4. Can previously unknown features be identified in the relevance of a given dataset?
5. Can outliers be identified, cost, and process effectively? Similarly, can redundant and irrelevant features be identified based on fitness factors?
6. Can features be grouped based on high correlation, relevance, and non-redundant scores? Can a model learn to do this on its own?
7. Can each participating feature (predictor) be weighted in a logical space where it's optimum-fitness can be measured?
8. Can Local gain (LG) and Global Gain (GG) constructs be programmed to quantify the predictive value of each participating feature?

9. Finally, can parallelism using 3D modeling be incorporated to enhance the classifier learning process?

1.4 Research Problem and Supporting influences

This research addresses three problems as reasoned below.

1. Evaluation of an algorithm is crucial for various classifiers. A general rule adopted by researchers is evaluating some of the models that algorithm generates[56]. Regardless of the underlying algorithm, every model is prone to overfit, underfit, low accuracy, or predictive errors. In line with the concept of "*No Free Lunch*" Theorem (Wolpert, 1996), several algorithms in *SL* are evaluated to find the best fit to the underlying data and predictive analytics in goal. This approach is not always cost and process effective nor can it ensure that a better algorithm is not left untested. Ensemble approaches including boosting, bagging, and staging have shown some performance overheads and susceptibility to overfit[57][58]. Therefore, research is open in the areas of active ensemble learning techniques specially in the relevance of improved performance measurements to improve the aggregation of weak and strong learners.
2. Determining the right number and the type of features out of the given dataset's attributes is also crucial[23][18]. It is not uncommon for the *ML* process to use dataset of available features without computing the predictive value of each[59]. Such an approach makes the process vulnerable to high computation cost, low speed, overfit, predictive errors, and poor generalization[60]. Each feature in the

dataset has either a unique predictive value, redundant, or irrelevant value[61]. However, the key to better accuracy and fitting for *ML* models is to identify the optimum set (i.e., grouping) of the right features set with zero redundancy and irrelevancy[20][23]. Therefore, as the saying goes "Less Is More" opens research opportunity to enhance the feature engineering in the relevance of the ensemble methods.

3. Corroboration and Performance of the *ML* models are the challenges of their own[62][29]. Traditionally, Accuracy is the most popular method used to authenticate the model internals[63]. Though, accuracy measure is still considered an important measure, the reliability of this measure specially in the relevance of 'Accuracy Paradox' is open for further research and investigation[64][65][66]. Accuracy and Error correlation often fail in balanced ratios. Therefore, performance metrics trade-off and baseline standardization for ensembles methods have rooms for further improvement.

Thus, based on above three research problem area, we formulate the main Hypothesis statement for this thesis as:

Enhancing the fundamentals of blending Machine learning methods via higher dimensions driven training schemes, learning from its mistakes (i.e., feedback), internal measurement, and features quantification optimize the model fit, integrity of the validation, and feature predictive value.

1.5 Scope of the Proposed Research

The research work contributed in this thesis falls under the zone of Data Science. The scope is limited to Supervised Learning ensemble, and can be outlined by the following highlights:

1. Ensemble techniques to fill the gaps as identified in the literature review screening[41][67][68].
2. Feature Engineering and Optimization in the dataset used for training the SL models[69][60].
3. Improving the training-testing (validation) of the datasets and performance measurement to corroborate the integrity of the model.[70][16]
4. Improving the model corroboration and performance measurements while addressing the accuracy paradox problem.

1.6 Performance Measure and Validation Approach

Engine validation and performance measures are vital to finalizing the engine's algorithms, internals, and APIs. Chapter 6 lists the various tables where we have quantified these vital measures. In some cases, these metrics do not correlate well either. Therefore, we have innovatively created a performance metric in our engine (i.e., *eWPM*) .*eWPM* is implemented inherently in the engine to measure the learning process using *LT* objects. We have reported the metrics in which we have obtained the stability of the results. As it is well known fact that *ML* classifiers and the metrics, that the engine is being evaluated on, may behave differently for different data sets[65][71]. For that very reason, we have

innovatively used a very diverse data sets (ideal for classification) so the metric correlation and trade-off can be obtained, and corroboration can also be obtained on the fair basis. In some of our tests, we noticed some of the metric such as Error and Accuracy were not correlating very well. For that very reason, we imposed error bounds (i.e., 80%>Error>20%) in the mathematical model to improve the fitness and real-world integrity of the outcome.

Also, it is vital to note that using specific data domain (e.g., *Health, Crime, Stock market, Academic, Epidemics,* etc.) were not required to develop and test the engine. Data from various domains were used in *SL* relevance. Therefore, we skipped listing the specific domains we used to develop the internals of the engine, as it is not relevant/beneficial to be known in this book. However, we have listed the specific data domains with details in our experiment section for evaluating the engine's (i.e., *when engine becomes a model*) external layers, in Chapter 5.

1.7 Quick Glance at Proposed Solution

eMLEE units are: *(i)* enhanced algorithm blend and tuning (*eABT*). *eABT* is based on parallel classifier learning and metrics check off. *eABT* unit is built of mathematical constructs and specialized functions (i.e., tuning and quantifying) governing the underlying internal *eABT*-algorithms. *eABT* unit aims to focus on *3D* modeling and measuring of the metrics by using internal scoring, classifying, and weighting functions. It objectively uses *x {0:1}, y {0:1},* and *z {0:1}* as *3D* coordinates to store the scores, which are regulated, parallelized, and stored by logical table (*LT*) constructs. *LT* constructs controls the entire process. These scores are used for creating and regulating optimum blend and maintain

necessary tune up while the training continues. During the tuning process, *eABT* is fed with wrong predictions, so it can also learn from its mistakes. The mechanics of this novel scheme greatly improve the learning process while ensuring the optimum blend in maximum fitting space, as affirmed by *3D* scoring. *(ii)* enhanced feature engineering and selection (*eFES)*[72] unit is built to weigh each feature so the optimized number and grouping is achieved. Each feature is computed in *3D* space for each sub-coordinate (i.e., *x=over-fitness, y=under-fitness, and z=optimum-fitness*). Such scoring in each dimension further helps to quantify each new feature for its optimum predictive value, as the group function further evolves. Such approach ensures optimum number of features in the final set to optimize the features handling; *eFES* is built using a unique scheme to regulate error bounds and parallelize the addition and removal of a feature during training. *eFES* also invents local gain (*LG*) and global gain (*GG*) functions using *3D* visualizing techniques to assist the feature grouping function (*FGF*). *FGF* scores and optimizes the participating feature, so the *ML* process can evolve into deciding which features to accept or reject for improved generalization of the model. *(iii)* enhanced Cross Validation and Split *(eCVS)* is investigated to propose the engineered value of *k* to determine the optimal split while the blend of the algorithm is being contrived. It must be noted that *eCVS* is at infancy of the research. *iv*) enhanced Weighted Performance Metric *(eWPM*) is constructed to ensure the improved correlation of the various *ML* metrics in the blended model. All the units are governed and regulated by the centralized unit known as Logical Table (*LT*) using built parallelism[73]**.** *LT* is constructed in the most inner (*i.e.,* the lowest) layer of the engine to coordinate the measures while the blend is in progress. The *LT* is an in-memory logical component, that governs the progress of *eMLEE*, regulates the engine metrics, improves

the parallelism, and keep tracks of each element of *eMLEE* as the classifier learns. Optimum fitness of the engine with parallel “check, validate, insert, delete, and update" mechanism in *3D* logical space is obtained.

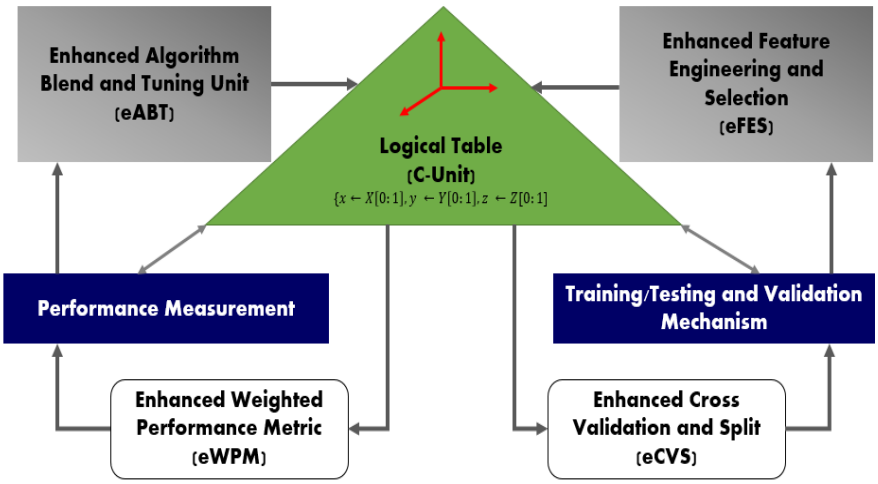


Figure 1.1– Conceptual Flow of eMLEE on elevated level.

1.8 Parallel Processing by Design

Parallelism of the hardware and software process have been implemented for decades to improve the efficiency and performances of several systems, machine learning, and framework[74][75][76][77]. We implement parallelism inherently in the internals of *eMLEE*, as detailed below.

As shown in **Fig. 1** for the block diagram of the *eMLEE*, the parallelism is built via the centralized unit (i.e., Logical Table). *LT* unit creates logical coordinates (*x, y, and z*)

for each method (i.e., *eABT* internals) and each feature (i.e., *eFES* internals) and updates the values as $x \in X(0:1), y \in X(0:1), z \in Z(0:1)$.

At the inner layers, the parallelism for *eABT* is done via the following steps:

a) Algorithms are either added randomly for the initial test and then once the classifier achieves the maximum accuracy, the *LT* object starts regulating the process by using Adder and Remover functions (described in the math model in detail), and logical rows are created to monitor and record the metrics of the classifier at it is trained and tested. Then each logical *3D* point in space evolves into optimum space, and the records are updated in the logical table.

b) Next, the *LT* object ensures the learning from its mistakes function is evaluated in parallel while the metrics are being measured and recorded. This way, the error bounds functions and gain functions ensure the accuracy and error correlation to its best scoring for improved generalization.

c) The *eWPM* metric computation is done in parallel at the lowest level when the *eCVS* engineers the optimum value of k, such that the following holds in all cases: $CV\{k = 1,2,3, \ldots . . n\} \cong eCVS \sim eCVS(k' \in k \leftarrow \{LT(z)\})$

At the inner layers, the parallelism for *eFES* is done via the following steps:

a) The available features $F(x, y, z) = \{f_i \in F\}$ are given a logical entry in the learning space by the *LT* handle to compute its predictive value {0:1} via the measurement of Irrelevancy and Redundancy functions (described in the model later), and entry is modified in the table while the test-train is being engineered by the logical space.

Thus, the model is pushed to the optimum space for the features that has non-overlapping correlating values.

b) The *eWPM* metric is monitored closely by the *LT* object so the range can be determined by the engine to ensure the improved trade-off while more features added in parallel. This is the best way, that we have experienced to minimize the overfitting that every model is prone to. This may be done sequentially, but parallelism improves the insurance that each feature is evaluated at the same time; the classifier is incorporating metrics reading from LT handle.

1.9 Novelty Highlights for the Technical Layers

The following points outline the highlights that are novel approaches to the best of our knowledge and survey.

1. In our experimental tests during the evolution of this research, we felt the necessity of a unit as a centralized part of this engine that governs, regulates, and keeps track on the underlying data based on improved parallel processing in *ML* process. The challenge of trade-off between vital metrics such as complexity, accuracy, speed, etc. becomes also very important and that is where *LT* plays a significant role. *LT* creates parallel process for each element in each run governed by *3D* object co-ordinates (*x, y and z*) and then makes observations in the real time of classifier learning and updates its logical row in the table.

2. Local and global error bounds definitions in *3D* space between 20 to 80 % for model to be realistically trained for diverse set of data to achieve improved generalization and accuracy.
3. Local and Global gain functions to correlate and aggregate the mix of methods in blended model for improvement in visualization, study, validation, and experiments of the proposed engine. This approach reduced the bias and overfitting of the model.
4. Developing Learning from Mistake (*LFM*) function.
5. Developing an optimized feature grouping function to improve pre-processing and reducing the cost of complexity and speed during training.
6. Evaluating and quantifying each feature in a dataset for its predictive, irrelevancy, and redundancy value for maximizing pre-processing computation, speed and outcome in predictive modeling.
7. Developing an aggregated and weighted performance metric suitable for blended model for improved *3D* validation and visualization of classifier learning.
8. Improving cross validation process via parallel processing for value of k. It should be noted that future research and experimental proof are needed to confirm this enhancement.
9. Developing generalized algorithms (1-8) that constitute the math model and architect the *eMLEE* engine as a black box. These algorithms can be further tuned and used via APIs for any sort of supervised learning-based prediction such as PAE (our implementation/applied model)[78][79].

1.10 Contribution Highlights

This work:

1. Contributed to 'No Free Lunch' solutions.
2. Improved the fundamentals of the Ensembles using *3D* scoring of the elements of the learners for improved quantification of the measures.
3. Improved the fitness space for overfitting and underfitting problems.
4. Improved the feature's predictive value quantification and measures.
5. Introduced a Logical Table construct (i.e., unit) to govern the underlying algorithms blending and feature engineering using parallelism and object-oriented approach.
6. Improved the performance measure approach of the engine (classifier learning).

1.11 Book Anatomy

Chapter 2 discusses the Literature review that resulted by years of work in the proposed research.

Chapter 3 presents the theoretical foundation and related mathematical modeling developed to solve the research problems, for major units of eMLEE engine, as eABT and eFES.

Chapter 4 presents the internal algorithms of the engine.

Chapter 5 presents the experimental setup, results and analytical discussion to support the validation of the engine for the contribution this thesis makes.

Chapter 6 provides the comparisons tables for both the major units of eMLEE with the most popular metrics used in the literature for newly developed engine, such as eMLEE.

Chapter 7 conclude the book with Final remarks, future works, limitations and future resources for the new researchers, industrialist and community, and for those how who will take this work towards further enhancement.

Reference and Appendices are provided at the end.

CHAPTER 2 – LITERATURE REVIEW

2.1 Survey Approach Explained

Study, knowledge, and comparison were crucial steps to the cause and foundation of our proposed work. Several reputable journals were searched for the latest published material in the relevance of the research presented in this book. We limited our scope to the supervised learning methods and then we conducted some research in the application of these techniques in some of the demanding areas such as academics, social networking and career predictive modeling. We were mostly focused on the progress of the ensemble techniques and their different versions to understand how they improved the common challenges as outlined in Chapter 1. Thus, we were able to identify gaps in each study and examine the challenges and opportunities from the existing knowledge and progress.

2.2 Context of Machine Learning

2.2.1 Supervised Learning Algorithm Engineering

With tremendous growth in data and associated new features, boosting[57], ensemble[80], and blending of *ML* algorithms have attracted attention of many researchers in the recent years[81]. To identify the gaps in the latest state of the art in the field of *blending*, we considered area of *ML* where *blending* was of high relevance. Predictive modeling, as our focus for the consumption of the proposed engine, is a great candidate to

be looked at, for such opportunities. One of the challenges in such research is to determine the algorithm that has shown relatively higher fitness than other algorithms or techniques. If such knowledge can be extracted and quantified, *ML* blending process can dramatically be improved. On the other hand, blending two or more algorithms can also reduce the learning performance if not evaluated or weighted in parallel. A recent noteworthy advancement includes MKBoost[67], SemiBoost[82], Bosco[83], StructBoost[68], and Ensemble learning[84].

Zhou, *et, al.* [85] proposed a new ensemble algorithm as Filtered Attribute Subspace based Bagging with Injected Randomness (FASBIR) mostly focused on K-NN classifier learning. Their proposed work outperformed the bagging techniques while doing the integration of the perturbations on the training data along with input features. However, their work did not address the effectiveness of multimodal perturbation in different kind of base learners. Sun, *et, al.*[86] proposed a method of adding ensemble of basis vectors all at once as compared to the existing techniques in sparse approximation to reduce the required number of forward steps and computational cost. This approached was also applied to a large classification task in which traditional selected-based method failed to work properly. However, their work still needs to be considered for more kernel methods such as *SVM, KLR*, etc. To improve the predictive analytics, boosting regression trees have shown some promising results. Bergstra, *et, al.*[87] research was focused on non-linear regression estimation for *ML* process. Their proposed method outperformed the traditional model-based, where model is only built based on hardware inner workings and empirical auto-tuning. Their work has room to extend towards adding more features specially in the

hardware domain. Also, predictive auto tuning can be tried to evaluate the extension of their work. Asadi *et, al.*[88] used gradient-boosted regression trees for their learning process in *ML*. Their work and experiment revealed that memory-based data structure and vectorization improved the exploitation of modern processor architectures. Their work needs to be tested for different architecture-based data sources. *ML* has shown the enormous potential in image processing, classification and recognition. Samat *et, al.*[89] extended the state of the art in areas of extreme learning machines (*ELM*) by considering the outcome of *ELM* for high-dimensional data, e.g., hyperspectral image. Therefore, they introduced an extension of *ELM* by proposing bagging based and Ada-boost based *ELMs* to improve image classification. Their results indicated that *BagELMs* and *BoostELM*s have outperformed *SVM* and had achieved improved classification performance. Their work can be used to consider other differential and non-differential functions along with comparison with other *ML* techniques and kernel methods. Another similar research in image classification has shown promising growth of *ELM* to improve the classification. Bencherif *et, al.* [90] proposed an extension of multiclass active learning (*AL*) method for sending image classification. They utilized the capability of *ELM* and graph-based optimization method to improve the classifier accuracy. Their work though used three different datasets, but more datasets specially with diverse feature sets need to be tested. Also, their work can further be tuned to utilize the blend of the algorithm where *ELM* can improve the speed of *AL* with minimization of complexity function and computing cost. A noteworthy progress has been made in the application of electrical engineering and biomedical industries. Alves *et, al.* [44] considered the computational biology area for predictive modeling. *PSO* has widely been used in metaheuristics and optimization

problems. Authors also proposed extension of the techniques as Multi-Swarm Ensemble (*MSWE*) to study the enhancements in the ensemblers. Their work showed the outperformance of *MSWE* as compared to each *ML* algorithm in their nine datasets for experiments. Their work has opened wide door to get into for further biometrics research and predictions specially to see more of unsupervised learning algorithms, that can further improve the learning process specially for unlabeled data in genomics domain. Tandon *et, al.*[91] discussed the importance of machine intelligence in big data domain towards natural language. Their work provided great motivation towards mining common sense that can be extracted from the words of people, but it did not provide in-depth analysis of algorithms or features that may impact such intelligence during learning process. Hernandez *et, al.*[92] discussed the parallel processing optimization in the big data applications. Their results showed improved recommendations score for resources and workload but did not address or consider the parallel processing of various algorithms to see if that could further improve their work. Dai *et, al.*[93] work was focused on multiple classifier systems (*MCS*s) with their contribution of supervised competitive learning algorithm (*SCL*) to improve the accuracy of the classifiers. Though their work showed satisfactory progress for accuracy measurements but did not consider other metrics of the supervise learning classifier especially if algorithm blend was intended. Tuia *et, al*.[28] provided survey of active learning algorithms in the field of remote sensing image classification. Mainly focused on *SVM* algorithm, they discussed the issue of efficient training set, having high impact on the expected outcome. Their findings, results, and discussion showed that active learning algorithms are making great progress especially for image classification and the type of data it involves. However, their contribution was

limited to active learning, especially for image classification and may not be suitable to apply for a diverse set of data and features. Garcia *et, al.*[94] provided a survey on discretization techniques with empirical analysis in supervised learning. Discretization is an important approach specially to improve the underlying algorithm in terms of feature/attribute tuning and qualitative analysis. They provided in-depth analysis and guidelines of various methods with taxonomy table of their findings. Their findings also suggested an ideal selection of a method for given problem. Their findings and experiments showed accuracy of various *ML* techniques but did not provide other metrics that may be of special interest especially when blend is being engineered for a greater generalization. Wang *et, al.*[95] discussed the process of purchase decision in subject minds using *MRI* scanning images through *ML* methods. Using recursive cluster elimination based *SVM* method, they obtained higher accuracy (71%) as compared to previous findings as per their research. They utilized Filter (*GML*) and wrapping methods (*RCE*) for feature selection. Their work though provided great foundation and motivation for feature processing but did not provide the in-depth experiments of application of the technique on neutral subjects where feature may mislead, and algorithm design must take this into account.

Some of the work in the areas of engineering domains such as antenna design, wireless communication, chip designs and other biomedical engineering are using advanced *ML* techniques with recent availability of digital data. Liu *et, al.*[96] addressed the low efficiency of evolutionary algorithms in Electromagnetic(*EM*) design problems due to the cost, and thus proposed a new method called surrogate model differential evolution for antenna synthesis using *ML* techniques. Their work was very limited to *EM* applications

and did not provide the wide applicability to other domains of similar challenges in *EM* or Electrical engineering domains. Yu *et, al.*[97] focused on weaknesses of semi-supervised clustering algorithms and to address these challenges, they proposed closure based constraint approach and random bases semi-supervised framework. They used datasets from medical domains such as cancer patients. Their work lacks dealing with pairwise constraints and removal of redundant constraints. Such limitation may be addressed by the work in the feature optimization and engineering as we propose. Xiao-jian *et, al.*[98] advanced the work in optimization extreme learning machine (*OELM*) for the error penalty parameter *C*. Their work extended the traditional *OELM* classifier with the regularized parameter *v*. Their work created useful foundation for classifier parameter optimization. However, they lacked to confirm the stability of optimization if different classifiers were used or tested.

2.2.2 Feature Optimization and Engineering

To identify the gaps in the latest state of the art in the field of feature optimization (*FO*), we considered area of *ML* where *FO* was of high relevance. In general, every ML problem is affected by feature selection and feature processing. Predictive modeling, as our focus for the consumption of the proposed model, is a great candidate to be looked at, for *FO* opportunities. One of the challenges in *FO* is to mine the hidden features that are previously unknown and may hide a great predictive value. If such knowledge can be extracted and quantified, ML process can dramatically be improved. On the other hand, new features can also be created by aggregating existing features. Also, two irrelevant features can be combined, and their weighted function can become a productive feature

with higher predictive value. In this section we provide the related study of noteworthy references and then list the gaps we identified. **Li *et al.*** [25] presented a detailed review of the latest development in the feature selection segment of machine learning. They provided various frameworks, methods, and comparisons in both Supervised Learning (SL) and Unsupervised Learning (UL). However, their comparative study did not reveal any development where each feature can achieve a run-time predictive scoring and can be added or removed algorithmically as the learning process continues. **Lara *et, al***.[99] provided survey on *ML* application for wearable sensors, based on human activity recognition. They provided a taxonomy of learning approach and their related response time on their experiments. Their work also supported feature extraction as an important phase of *ML* process. Their work provides great motivation for feature engineering and further improvement in feature selection and optimization. **Vergara *et, al***.[100] reviewed feature selection methods. Authors presented updates on results in unifying framework to retrofit successful heuristic criteria. The goal was to justify the need of feature selection problem in-depth concepts of relevance and redundancy. However, their work lacks to address the issues of model fitting when a diverse set of features are involved in datasets. **Mohsenzadeh *et, al*.**[101] utilized a sparse Bayesian learning approach for feature sample selection. Their proposed relevance sample feature machine (*RSFM*), is an extension of *RVM* algorithm, previously invented. Their results showed the improvement in removing irrelevant features and producing better accuracy in classification, better generalization, less system complexity, reduced overfitting and computational cost. However, their work needs to be extended to more SL algorithms. **Ma *et, al***. [16] utilized Particle Swarm Optimization (PSO) algorithm to develop their proposed approach for detection of falling

elderly people. Their proposed research enhances the selection of variables (such as hidden neurons, input weights, etc.) The experiments showed higher sensitivity, specificity, and accuracy readings. Their work though in the domain of healthcare industry does not address the application of approach to a different industry with an entirely different dataset. **Lam *et, al***. [17] proposed a unsupervised feature-learning process to improve the speed and accuracy, using the Unsupervised Feature Learning (UFL) algorithm, and fast radial basis function (RBF) for further feature training. However, the UFL may not fit when applied. SL. **Han *et, al***.[104] used circle convolutional restricted Boltzmann machine method for 3D feature learning in unsupervised process of ML. The goal was to learn from raw 3D shapes and to overcome the challenges of irregular vertex topology, orientation ambiguity on the surface, and rigid transformation invariances in shapes. Their work using 3D modeling needs to be extended to SL domains and feature learning. **Zeng *et, al***.[105] used the deep perceptual features for traffic sign recognition in the kernel extreme learning machines. Their proposed DP-KELM algorithm showed high efficiency and generalization. However, the proposed algorithm needs to be tested across different traffic systems in the world for more distinctive features than those they have considered. **Wang *et, al***.[95] discussed the process of purchase decision in subject minds using MRI scanning images through ML methods. Using the recursive cluster elimination-based SVM method, they obtained higher accuracy (71%) as compared to previous findings. They utilized Filter (GML) and wrapping methods (RCE) for feature selection. Their work also needs to be extended to other image techniques in healthcare. ML has also shown a promising role in engineering, mechanical, and thermo-dynamic systems**. Zhang *et, al***.[106] worked on ML techniques to do the prediction in the thermal systems for systems components. Besides

many different units and technique adoptions, they also utilized FS methods based on correlation feature selection algorithm. They used Weka data-mining tools and came up with the reduced feature set of 16 for improved accuracy. However, their study did not reveal how exactly they came up with this number and whether different number of the features would have helped any further. **Wang *et, al***.[107] used the supervised feature method to remove redundant features and considered the important ones for their gender classification. However, they used the neural network method as a feature extraction method, which is mostly common in unsupervised learning. Their work is yet to be tested for more computer vision tasks including image recognition tasks in which bimodal vein modeling becomes significant. **Liu *et, al*.**[108] utilized the concept of F-measure optimization for FS. They developed a cost-sensitive feature approach to determine the best F-measure-based feature for the selection by ML process. They argued F-measure to be better than accuracy, for purposes of performance measurement. However, accuracy is not sufficient to be considered a baseline for performance reflection of any model or process. **Abbas *et, al***.[109] proposed solutions for IoT-based feature models using the multi-objective optimum approach. They enhanced the binary pattern for nested cardinality constraints using three paths. The second path was observed to increase the time complexity due to the increasing group of features. Though their work was not directly in ML methodologies, their work showed performance improvement in the 3rd path when the optional features were removed.

2.3 Context of Predictive Modeling for Academia/Career

2.3.1 Recommender Systems

The new era of social networking and digital ways of creating and expanding networking in today's world of the web has provided with many opportunities to the researchers to build various recommender systems in the last two decades. A study done by [110] builds social network based recommender system model and compared with collaborative filtering and emphasized on three factors that influence pretty much any kind of decision we make in our daily lives. i) User's own preference, ii) public reviews or comments iii) friend's or fellow recommendation. Their model produces 17.8% better results than Collaborative filtering (CF). Authors in [111] uses Facebook data to build recommender system to choose friend based on social graphs. Authors in [112] utilizes location of user's and their preference to improved recommendations and suggestions. Authors in [113] presents recommendation system for researchers to find their related work using collaborative filtering based approach and create personalized approach to find related article of research. Their system, even recommends the researcher when she or he is not doing active search on web. Authors in [114] use decision trees to collect past student data and build a recommendation system to help choose relatively better academics journey. They also emphasize on behavioral analysis of individuals to contribute to such recommendations and decision-making process. A study done by authors in [115] uses data mining techniques to predict student performance and develop recommender system. They compared with regression methods and logistic regression by utilizing educational data for their system build up. Personality prediction research, mining and analytics have helped research community to see through the lens of how individuals exhibit themselves and how they fall into various categories of Big Five personality traits[5][6]. Such knowledge helps

to create segment of such individuals for various industries including academics. Machine learning techniques applied to social networking data of Facebook in study done by authors in [116] uses data set of 537 individuals to predict Big Five model traits for various cases. Similar study done on Twitter data[117] where they train ZeroR and Gaussian Process to predict scores of personality traits.

2.3.2 Retention

Data mining and statistical methods have been used to predict student retention in the colleges. Dey, et al.[118], discuss five categories about prediction such as achievement, demographic, financial, social, and psychological. Authors further discuss the impact of Big Five factor model traits on retention either directly or indirectly. Chen, et al., explore the opportunities of big data for student prediction and retention using machine learning supervised learning and regression classification method. They perform case study on their students in group for predicting in collaborative learning[119]. Their work progresses intelligent recommendations for individuals before and after coursework for future success and retention as one of the predictor. Alkhasawneh, et al., utilizes neural networks to train data for modeling student retention in science and engineering majors[120]. Out of their 338 samples, 70.1% students are classified accurately by using two NN networks. They focus on GPA to predict and classify the students into three categories as at-risk, intermediate and advanced. Hadda, et al.[121], raise an important research question about if computation thinking and related course taken by student can predict their performance for future. They use current GPA in the course and then accumulative GPA on data set of 982 students taking computational thinking courses in 2-year study program. Statistical

analysis of their model provides about 95 % of confidence, finally. Sarkar, et al.,[122] argue against the complexity and inflexibility of traditional survey-based approach for student retention modeling and support neural network to show improved results in their work. Sharabiani, et al.[123] uses database of undergraduate students at university of Illinois at Chicago and predict student grades in three courses for their second semester. They utilize Bayesian networks framework to develop their model and they test their model to outperform existing models in their study. Slim, et al.[124] uses linear regression and Markov Models to build prediction model for data set of 41,498 currents students and about 400 new graduates at University of New Mexico and predict early semester with minimized error rate (0.0449) as compared to existing techniques.

2.3.3 Academic and Career Prediction

Various case studies[125][126] support to predict drop outs and improve student performance. Student behavior[127][128] prediction plays role in such type of data mining and conclude predictive metrics and measures. Student Recommender system[129] has matured through various researches in student data mining[130][131]. Recent research in data mining and data analytics on unstructured data has shown great potential. LinkedIn has played great role to accumulate such data. Many Researchers have used sophisticated data mining algorithms to improve predictions and extract useful knowledge that has helped the industry in recent years. Another study[132] discusses Prescriptive, Predictive and Descriptive Analytics using Machine learning algorithm and other advance tools. They support data analytics to help understanding customer, Risk analysis, Finance and

workforce. Research done in [133] shows military training using data analytics. Personality Prediction from Facebook and Twitter in correlation with LinkedIn data analytics improve such analytics dramatically[134][135] [136]. These researches use data mining algorithm such as Support Vector Machines, Bayesian Network and K-means. Five Factor Model (FFM) also known as Big Five Model has been used to predict personality on general level using five traits [137][138][139][140]. However due to limitations in this model, granular level of prediction such Good Fit Candidates can't be done using this model with great reliability. In online education system, the availability of student information is huge. Research shows, students' data can be clustered to assess by algorithms and predict their performance[141]. The application of Big Data and Learning Analytics can analyze large-scale academic data and produce results that can transform the educational system to fit to the right students[142]. The Spectral Clustering model can predict student performance by exploiting data cluster[143]. Moreover, Correlation Based Feature Selection algorithm can predict students' grades by evaluating student enrollment data and results of previous examinations[144]. Research identifies, the student performance evaluation can be studied in two patterns 1) overseen and unsupervised environment and 2) students' performance in mathematics, English and in other programming courses[145]. Therefore, the evaluation can be helpful to design new evaluation process, rearranging new academic syllabus and prerequisite courses. Recommender systems are being used to suggest most suitable courses, curriculum, modules and personalized knowledge components. This helps to maximize the students' learning abilities[119]. Data can be clustered based on the clustering methods such as centroid based, density based and distribution-based clustering. These data mining techniques can be applied to study students' related data of primary,

secondary and higher level education systems[130]. Machine learning and its prediction power[116][146] have emerged and evolved into many sophisticated algorithms. A study done in [147]machine learning technique based on Rough Sets to extract rules for their prediction work. They claim to have better efficiencies as compared to SVM (Support Vector Machine) methodologies[148][149]. Various comprehensive reviews[5][6] on Search of individual personality types for constructive real world application has been conducted. They support the future mining of data to address challenges and issues, at hand. Similar to data on social networking that contributes to Personality Prediction Opportunities and research, Educational data mining Big data Education[119] contributes to prediction and classification of factors in huge data set for success and failures of students and educational system in context. A stream of research work[142] [150] has supported and advanced the techniques and data mining[151][152] algorithm to improve prediction accuracies. Various case studies[125][126] support to predict drop outs and improve student performance. Student behavior[127][128] prediction plays role in such type of data mining and conclude predictive metrics and measures.

2.4 Context of Social Networking

2.4.1 Personality Computing and Prediction

Personality study has been a topic for decades for research in psychology and social sciences. However, with recent era of internet and social networking platform such as Facebook and Twitter have revolutionized the researcher's ability to study personalities and envision the applicable usage in the real-world. A survey done by authors in [6] stresses

the personality computing applications with discussing related technologies. They conclude with focus on improving machine learning algorithms to create better models and integration of human sciences and computing to better utilize the personality features for better predictions for various applications. Study done by authors in [153] study measures from social behavior to predict personality. They use Gaussian Processes and Zero R with Five Factor Model (FMM). Other numerous studies [154][155] utilizes behaviors and smart hubs to predict personality, mainly form social networking data. Twitter and Facebook researches, such as in [156] authors addresses personality that is result of people interacting and communicating in Facebook. Machine Prediction in study done on Facebook profiles to rank individuals using Big Five Model[116]. A study done in [157] finds the relationship of Facebook popular users contacts in real world to their electronic world. Twitter data has widely used to predict personality and traits based on what they tweet and other relevant posts. Similar to the data on social networking that contributes to Personality Prediction Opportunities and research, EDM and Big Education[119] contributes to prediction and classification of factors in huge data set for success and failures of students and educational system in context. A stream of research work[142] [150] has supported and advanced the techniques and data mining[151][152] algorithm to improve prediction accuracies.

Five Factor Model (FFM) (OCEAN) [158][159] is widely used in research to predict and classify the various individual based on their behavior. A numerous studies [155][160] have shown the characteristics of data mining and personality traits such as FFM with Social networking data, to be very useful to categorize personalities accordingly.

A study in [139] discusses the measurement and Big 6 and Big 2. One of the challenges in FFM is different interpretation of its measurements that introduces some complexity. A Study[161] prepared on Big 5 Inventory Report (BFI) for better assessment of personality traits. One of their focuses is on language translation for individuals from many other nations. They include understand and verify if FFM model is standing well across various cultures and validity of individual's profiles in such various nations. Social Media including but not limited to Facebook, Twitter, LinkedIn, flicker, Pinit, Google+ and various blogs have great tendency to capture and store the personality features that are indirectly hidden the post user generates on daily basis, in various context of industry and discussion threads. A study done in [153] the examination of behavior traits in predicting personality and analyze to what extent is the relevance. They utilize Facebook and Twitter data as test data. They conclude the same success in prediction personality from behavior analysis as compared to results that can be achieved using text analysis. They also take the friends and follower behavior into consideration. A study done in [137] includes 444 users for their test. They utilize real time online individuals' behaviors to come up with algorithm using regression technique in data mining. They showed improved accuracy in relevance of related study and provide great reflection of online attitudes. They suggest continuing work in direction of other parameters like mental state and social networking behaviors. The Five Factor Model (FFM)[139][161] of personality uses personal data to classify an individual into one of the OCEAN personality traits, classifications that can subsequently be exploited by business entities. The fascination of firms with personal data has seen a surge in the number of blog sites and news posts from various organizations[162]. However, further understanding of such data to extract specific knowledge and

understanding of the industry is still an open research area. The OCEAN model[137] analyzes the personality of technology users based on the five basic foundations which include openness, conscientiousness, extraversion, agreeableness, and neuroticism. While the five-factor model is used widely, it exhibits some challenges. For instance, the elements lack independence from each other. Some facets show a negative correlation as noted between neuroticism and extroversion. Also, the model fails to explain all human personalities. Some of the features that the OCEAN model fails to consider include religiosity, gender, conservativeness and honesty [125]. Therefore, the sole reliance on data companies on the FFM traits model to understand user personalities is inaccurate. However, an integration of the FFM and personality features can provide an accurate reflection of the aspirations and objectives of an individual. Such proper understanding can be applied in guidance and counseling to enable learners to make good career choices. Besides, the integrated data model can be exploited to facilitate growth at personal and institutional levels.

Micro-blogging and predicting the personality is hot research area. A numerous studies [155][160][138] have shown the characteristics of data mining and personality traits such as FFM with Social networking data, to be very useful to categorize personalities accordingly. Machine learning and its prediction power[116][146] have emerged and evolved into many sophisticated algorithms. A study done in [147]machine learning technique based on Rough Sets to extract rules for their prediction work. They claim to have better efficiencies as compared to SVM (Support Vector Machine) methodologies[148][149]. Various comprehensive reviews[5][6] on Search of individual

personality types for constructive real world application has been conducted. They support the future mining of data to address challenges and issues, at hand. Similar to data on social networking that contributes to Personality Prediction Opportunities and research, Educational data mining Big data Education[119] contributes to prediction and classification of factors in huge data set for success and failures of students and educational system in context. A stream of research work[142] [150] has supported and advanced the techniques and data mining[151][152] algorithm to improve prediction accuracies.

2.4.2 Educational Relevance in Social Domains

Social Networking is the new trend to stay connected virtually with others. The growth of online social networks (OSNs) is huge. The number of users and their increasing list of friends can be studied by graph theory and log analysis tools[163]. Research shows, Lexical variety, Sentimental analysis and clustering analysis can analyze the text data. This can be helpful to analyze essays of students and predict their academic areas of interest[164]. In the field of education, social networking sites have advantages and disadvantages. Lack of privacy, less authenticity in friendship, time consumption and miscommunication are some of the critical challenges. On the other hand, flexibility of learning, easy accessibility, repetitive and convenient are the major benefits of social networking sites[165]. With the advent of Information Technology, the number of decision support systems is increasing. For instance, text analytics and mining based DSSs, recommender and advisory based DSSs, internet of things based DSSs and so on[166]. Social networking sites data can be analyzed to identify the personality of the users'. Their

profiles, personal data sharing, interaction and activities on social networking sites can be studied to know which Big Five personality model is applicable to a specific user[167].

Many educators believe that social networking sites are beneficial for education. Many students are learning from social networking sites. Their interaction and connections with other educators makes it possible to share the knowledge. Webinars and showing education videos in YouTube and in Teacher Tube are advantageous to the students and educators[168]. According to authors, classification, latent knowledge estimation, Domain structure discovery, Network analysis, Relationship mining etc. are some of the educational data mining methods[169]. Authors in[10] have extended the previous research about profile pictures and personality impression. They looked at relation between profile picture selection and message user is intending to the world through their profile on Facebook. Their survey and feedback from sample users as chosen concluded that users are aware of the importance of selecting the type of profile picture they keep for short or long term. They found User's personality traits that had an influence on the picture choice they make as profile picture. For example, extraverted users select more self-representative photos and narcissistic users select more physically attractive pictures. Work done by[170] reveals that users wall and newsfeed are important segments to investigate and research to further understand personality patterns and self-presentation. They categorized self-presentational information on wall and self-presentational behaviors at news feed, to research personality traits and study their inter-relation. Study in [171] supports the potential of big social data to predict a five factor model of personality. They cited the relevant study done to indicate the accuracy of personality prediction is in moderate range with typical correlation between

the prediction and personality is in the range of r = 0.2 and r = 0.4, where r = reliability. Authors in [172] shows significance work done towards distinguishing personality types (out of Big five) for both popular users and influential user's posts. Their study supports that popularity is linked to imaginativeness and influential users show also organized behavior. They use three counts or parameters in twitter data set as i) following, ii) followers, iii) listed counts. They show that root mean squired error below 88 % for prediction of user's five personality traits in active user status. They argue that privacy data access is still a hurdle in improving accuracy and remains an open problem. They propose for future research three important directions, Marketing, User Interface design and Recommender system, based on personality traits that are revealed by studying user's data.

Study in [153] supports predicting personality with social behavior only in light of Big Five traits. Authors outline features of user behavior with the following groups. Network Bandwidth(NET), Message Content(MSG), Pair Behavior(PAIR), and Reciprocity of actions (REC), Informativeness(INF) and Homophily(HOM). Their results verify that personality can equivalently be predicted using behavior features as with text features. A detailed survey on Personality Computing in [6], elaborates on Automatic Personality Perception, Automatic Personality Synthesis and Automatic Personality Recognition. A work done in [173] shows that two important traits of personality as conscientiousness and agreeable predicted less dishonesty in academics. Though their meta-analytic results were limited to small number of studies, did contribute to a better understanding of factors that influenced such behaviors in academics. Their study opens a wide door to pursue further research to better understand more personality features to find

out potential in students in academic world towards non-ethics, such as cheating, dishonesty, etc.

2.5 Gaps Identified

The following points highlight the gaps investigated by the related study and our in-depth literature review for ***LT*** unit development. That further motivated our research to fill them.

1. "No Free Lunch" theorem is still an open research area. Related applied solutions and newly developed models are still at their infancy and provides numerous research opportunities.
2. Ensemble based ML models have lots of room for improvement such as Overfit challenge.
3. Accuracy paradox and metric trade-off are open for further enhancement in the underlying fundamentals.
4. Though feedback is not a new approach but learning from the mistake approach is not really found in the existing work.
5. Hidden feature relationships discovery is very open research areas.
6. 3D modeling of the internal metrics of the model is rarely found in the literature.
7. Feature quantification to observe the predictive value is rarely done via internal metrics correlation.
8. "Les is More" as the saying goes in Feature Engineering still provides wide gaps to fill.

CHAPTER 3 - THEORETICAL FOUNDATION OF THE PROPOSED RESEARCH

3.1 Introduction

eMLEE at its fundamental level is based on stochastic thinking. Our end goal was to engineer a system (black box) that works smarter rather than harder. Thus, we innovatively adopted the approach of high dimensional computation to observe, quantify, and record the elements of blending the methods and the features during classifier learning so we could constructively optimize the blend with right ingredients (i.e., with high predictive elements in the blend). Internally, the engine creates several models before it decides on final ones based on stochastic thinking. So, in other words, we make our engine learn like human experience and promise us the best outcome it can produce (i.e., high fit to the reality of the world from the data it is given to be trained with). Inspired by ANN method, *eMLEE* learns to adopt to data with diverse set of features types (i.e., rate of noise and outliers) to minimize predictive error and overlearning.

eMLEE, as a proposed engine consists of the three units and two supplementary constructs

- Logical Table *(LT*) C-UNIT as a centralized component for all the units of eMLEE
- Enhanced Algorithm Blend and Tuning (*eABT*) – UNIT A

- Enhanced Feature Engineering and Selection (*eFES*) – UNIT B
- Enhanced Weighted Performance Metric (*eWPM*) – CONSTRUCT 1
- Enhanced Cross Validation and Split (*eCVS*) - CONSTRUCT 2

The following subsections provide the foundation of the implementation of first two units. eWPM and eCVS are implemented implicitly in the entire engine. The detailed mathematical modeling of eWPM and eCVS is left for future works.

3.2 (C-UNIT) Logical Table Centralized Unit

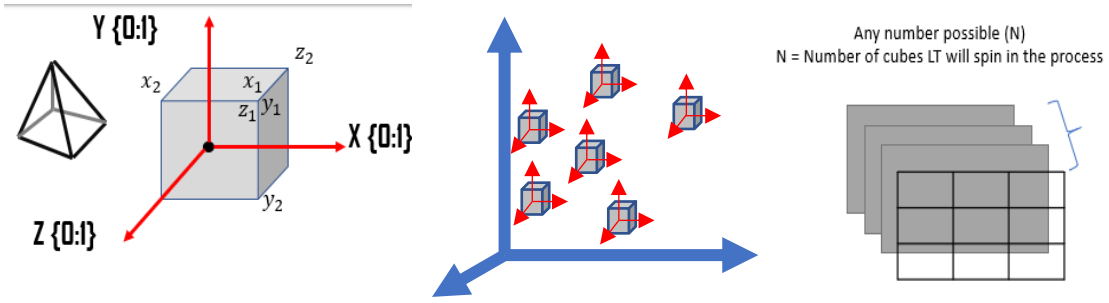
LT as previously discussed, is a vital central unit of *eMLEE*. *LT* is based on *3D* concept of optimization to regulate the metrics and learning progress of the engine. LT can be considered the backbone of the stochastic process takes place during learning stages and can be thought as CPU of the computer. However, unlike CPU which is a mathematical based architecture, *LT* is a mix of mathematical constructs and stochastic thinking and it learns to decide on its own to regulate and govern the *eABT* and *eFES* units of the engine. Thus, its provide Interface on an abstract level mimicking the *OOP* paradigm to promise optimized training experience for the learners as participants from *SL* pool.

3.2.1 LT Conceptual Illustration

Fig 3.1 shows three illustrations. (a) shows the cube creation with three coordinates as x representing the overfit, y representing the underfit and z representing the optimum fit. As (b) shows that LT handle creates several such logical cubes in parallel to search for the optimized one during elements from methods (i.e., SL algorithms) and features set. (c)

shows the logical structure of 2D table reference by another instance making it 3D array structure so LT can keep track of the internal measuring metrics during learning process.

Figure 3.1(d) shows the LT high level working principle.



(a) (b) (c)

Figure 3.1– Illustrating the Conceptual vision of LT internal working principle

3.2.2 Mathematical Constructs and Related Theory For eABT Unit Governance

LT operates in the memory and is dynamically updated. It keeps tracks of the methods (i.e., *ML 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. *LT* stores data based on three dimensions, where 'x' = *over-fitness,* 'y' = *under-fitness and* 'z' = *optimum-fitness*. In our design, we will be using "-*ness*" to mathematically phrase the metric for modeling purposes. At this stage, we refer fitness to be the overall performance of the engine, and our goal is to reduce 'x' and 'y' to as minimum as zero and improve 'z' to the highest possible value. '$\mathbb{R}$' is the ratio between the single error from an algorithm and

averaged error of all the algorithms in the blend.' $\frac{1}{N_e}$ 'is the normalization factor for the error *'err'*. *'Err'* indicates the Overall error determined for engine working phases.

$$\mathbb{R}_{eABT} = \frac{1}{N_e}\left(\sqrt{\frac{err}{err + Err}}\right)^2 \quad (1)$$

$$k_{x,y,z} = |\mu|^2 + \mathbb{R}_{eABT} \quad (2)$$

Where, μ computes all the values of *x,y, and z* components during learning.

$$\mu = \frac{1}{N}\sum_{i=1}^{N}(x,y,z)_i \quad (3)$$

Definition LT.1 – *Let there be a Adder Function as 'AddFunc(*$A(x,y,z))'$*, that adds each algorithm in the blend being processed, with Scoring Function as 'ScoFunc(0:1)' for each dimension in 3D space. Let there be a Remover Function as 'RemFunc(*), that must hold at-least one element per each test. * indicates the computed dimension.*

Construction - *LT* structure uses the grouping and scoring module. Scoring is based on binary number weights and based on the following rule.

RULE LT.1:

If *(LTObject.ScoFunc(A(i) > 0.5)* ***Then***
Assign "1"
Else
Assign "0"

By combining Gauss-Markov and Chebyshev methods[174] we construct adder function as given by

$$AddFunc = (A_n \cup A_{n+1})\left[\frac{\prod_{LT>0.5}\overline{\overline{A(z)}}}{BIN(min(x,y)}\right] \quad (4)$$

Our rule of thumb was 0.5 or 50 % to see how the engine learns. This way, we can separate the zone of over learning and under learning from a border line of 50 %. Once

classifier learns the zoning limits, it will decide this number itself. $\prod_{LT > 0.5} \overline{\overline{A(z)}}$ acts a regulating factor that provides the continuous product for each value of *z-dimension* for which the *BIN* function returns the least possible value of *x* and *y*. The removing function is given by

$$RemFunc(*) = (A_n \cap A_{n+1}) \left[\frac{\prod_{LT < 0.5} \overline{\overline{A(x,y)}}}{BIN(max(x,y)}\right] \quad (5)$$

It is imperative to validate the Adder and Remover functions at this point, using well known technique of Frobenius norm[175] form:

$$\|M\|(x,y,z) = k_{x,y,z} \sqrt{\sum_{x=1}^{X} \sum_{y=1}^{Y} \sum_{z=1}^{Z} M_{x,y,z}^2} \; (AddFunc, RemFunc) \quad (6)$$

Where M shows the matrix and we will elaborate on it in the examples section

Definition LT.2 – Let lt.Err be the specialized error function that implements the rule of optimum fitness (RoOpFit) as 0.2 < lt.Err < 0.8. Every entry in LT must adhere to this rule.

Construction – We construct very important error functions and rule of optimum fitness via Rule LT.2 as given by:

RULE LT.2:

Except random errors, lt.Err must be regulated to stay in between

20 to 80 % to avoid over and under learning.

If *(*lt.err < 0.2*)* ***Then***

Label it ‘Overlearning’

Elseif **(***lt.err > 0.8)* ***Then***

Label it ‘underlearning’

From the literature, we can implement the RMSE function for error determination, thus, we use our rule to build:

$$max(e{:}\,0.8) \;=\; \frac{1}{E}\sum_{i=1}^{Ne}\{(RMSE_i) \;-\; (100 \;+\; 0.2)/E\} \tag{7}$$

$$min(e{:}\,0.2) \;=\; \frac{1}{E}\sum_{i=1}^{Ne}\{(RMSE_i) \;-\; (100 \;+\; 0.8)/E\} \tag{8}$$

Using (7,8), we build the RoOpFit to lead towards determination of lt.Err function. RoOpFit regulartes the error that LT object can trigger for each test. Using kernel density function [176] and margin limits in Lipschitzness[13], we build

$$RoOpFit = \max_{err} < 0.8 \sum_{i,j}^{x,y}(A_{i,j}) \;-\; \min_{err} > 0.2 \sum_{j,i}^{y,x}(A_{j,i}) \tag{9}$$

With this error function being constructed, we can easily see the divergence in the optimum zone of z-axis. As discussed before, the *LT* object reads the previous entry and then based on the data from the training blend classifier, it updates (i.e., writes or deletes) in its logical structure (i.e., new or existing row of records).

Definition LT.3 – *Let* $\mathbb{B}_{A_n}$ *be a blending function and* $\mathbb{T}_{A_n}$ *be a tuning function that LT object must compute (detailed in algorithm definition).*

Construction - In *SL*, the classifier function $Classifier(\Delta S(x,y,z)) = \frac{1}{N}$ as identified, where $\Delta S = \{(i_{1,}o_1), (i_{2,}o_2), \ldots\ldots (i_{k,}o_k)\} \in (\mathbb{I}(input) \times \mathbb{O}(output))^k$. Where: $\mathbb{I} \subset \mathcal{R}^d$, and for regression: Errors $o_k \in \mathcal{R}$ For Classification: o_k is a discrete value. In Linear

Classification, as generally done in *SVM* concept: we can use Lagrange multipliers [177] to present the problem in equivalent maximization on γ:

$$\gamma = argmax \sum_{k=1}^{N} \gamma_k - \frac{1}{2} \sum_{k,l=1}^{N} \gamma_k \gamma_l (o_k o_l < i_k, i_l >) \tag{10}$$

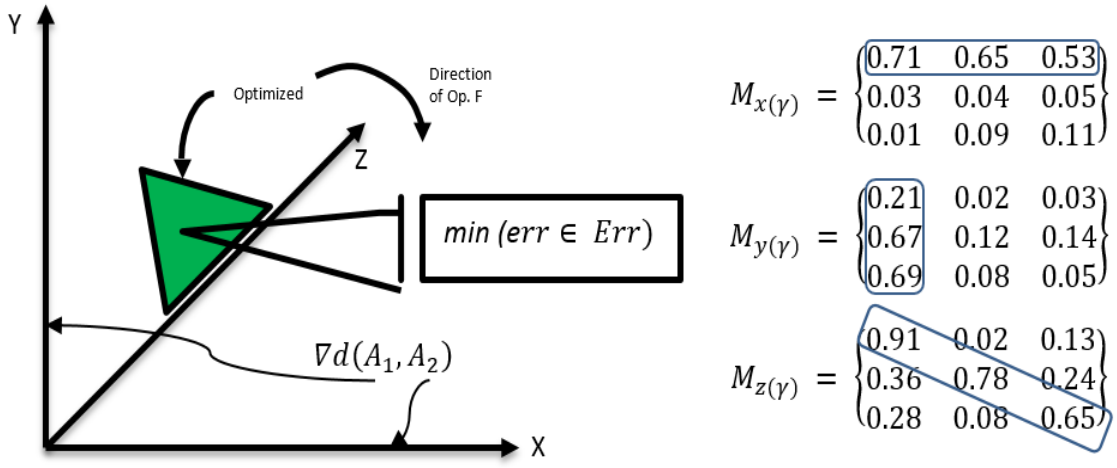


Figure 3. 2 - Illustration of optimum fitness logical *(x, y, z)* points

In **Fig. 3.2**, triangle is lying on z-axis, with the direction of momentum as being engineered in the model.∇d shows the Euclidean distance between two algorithms under test. The three Matrices shown are typical values for the sampling of the several hundred experiments. The encircled values show the optimized value of each axis for the desired optimization as LT object stores and reports. Equation (11) shows that Blend function is composed of three parts that work on *AddFunc*, *RemFunc* and score for each algorithm in each dimension as LT computes (See Algorithm 1 definition). Using Regularization in local minima where error is minimum but lipschitz loss[13] is unknown, we use vector product to keep the uniformity at minimum random distribution such that $z \neq 0\ AND\ x, y < 0.5$, thus, we construct

$$\mathbb{B}_{A_n}(0:1) = \prod_{k=1}^{AddFunc(k)} A_k \times \begin{pmatrix} x_k \\ y_k \\ z_k \end{pmatrix} + \prod_{m=1}^{RemFunc(m)} -A_k \times \begin{pmatrix} -x_m \\ -y_m \\ +z_m \end{pmatrix} \quad (11)$$

Tuning function is constructed using *Err* and *err* functions. As we stated earlier the *RoOpF* must be followed for blend to be tuned for improved optimization. *LT* object ensures by recording and manipulating the metrics, as per algorithm structure, discussed later. Thus, we can write:

$$\mathbb{T}_{A_n}(0:1) = \frac{1}{N}\sum_{i=1}^{N}(\mathbb{B}_{A_n}) - \|((Err - (Err + err)^2)\| \quad (12)$$

Definition LT.4 – *There must exist a cost function as ltCost, that must adhere to the minimum distance required between two algorithm during test, in logical space for ltCost(x,y,z), for which the condition ltCost(00,z)*$\in \Delta\,(Distance(x,y,z) > 0$, *always exist.*

Construction - During recognition of hidden patterns or points in datasets, the loss or cost function (*C*) is computed as

$$C(f(i: input), o: output)) = \frac{1}{2}|f(i) - o|, i_k \in \mathbb{I}, o_k \in \mathbb{O} \quad (13)$$

LT is built on three constructs: i) to monitor and store the ratio $\mathbb{R}_{eABT}$, ii) to update the values of x,y, and z components of each algorithm classifier during training, and iii) to score the algorithm $A_n|\{0:1\}, n \in (N+1)$, using Blending Function $\mathbb{B}_{A_n}(0:1)$, and Tuning Function $\mathbb{T}_{A_n}(0:1)$.

$$LT.eABT^{\boxplus} = \mathbb{R}_{eABT} \times \sum_{n=1}^{N} A_n(f(x,y,z) \left| exp\left(\frac{\mathbb{B}_{A_n}}{\mathbb{B}_{A_n} + \mathbb{T}_{A_n}}\right)\right| \quad (14)$$

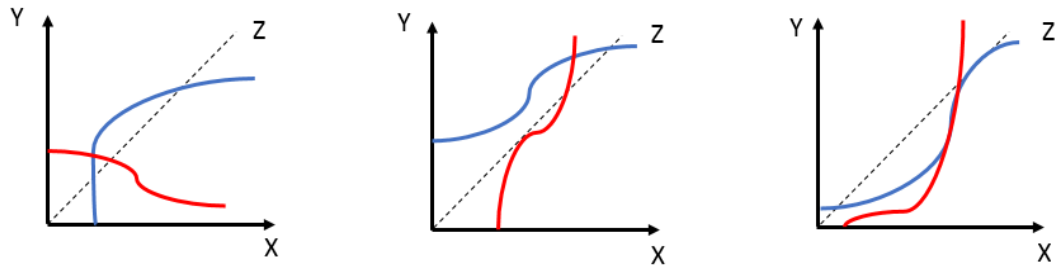


Figure 3. 3 - Illustration of $\mathbb{B}_{\mathbf{A_n}}(\boldsymbol{Blue})$, $\mathbb{T}_{\mathbf{A_n}}(\boldsymbol{Red})$ as they theoretically spread in optimum space of x, y, z dimensions

Fig. **3.3** shows three adjacent visual concepts. Our goal is to optimize the Blending and Tuning function with LT objects such that, it corresponds to high convergence in z-dimension. As shown in the **Fig. 3.4**, the values are updated based on the function that we built using simple linear regression, so when we fit a line on the given points, we can estimate the linearity of the classifier that is being built by the model as more methods are blended (governed by $\mathbb{B}_{A_n}(0:1)$) and then tuned (governed by $\mathbb{T}_{A_n}(0:1)$). **Fig. 3.4** shows the internal mechanics of the *eABT LT* working unit. It is internally based on binary classification technique. As the logical table grows with the quantized output as explained above, it decides which algorithm is a good fit in the blended model. *LT* governs the process at the lowest level of the engine being proposed. It creates the entry for each dimension *(X, Y, and Z)* as shown. As a threshold, if the *LT* value is less than 0.5, it is assigned binary '0', and if it is > 0.5, it is assigned binary '1'. Based on this, the binary truth table is built, and is used in the algorithm 1.

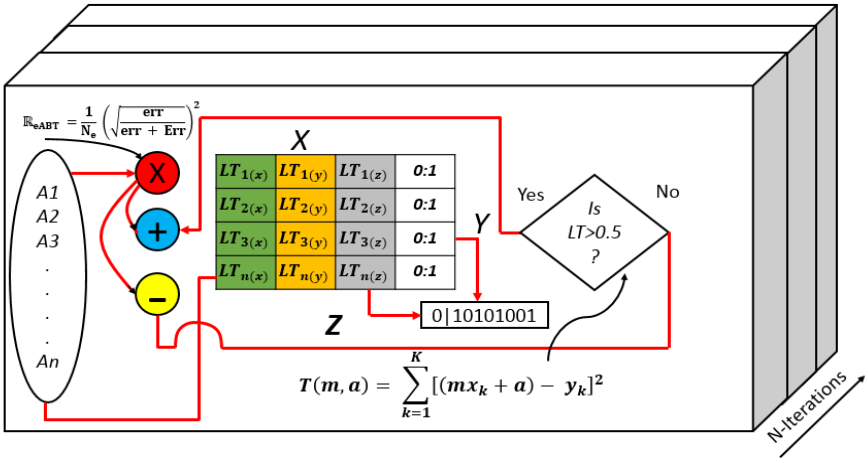


Figure 3. 4- Illustration of *eABT* Logical Table Internals

Based on illustration in **Fig. 3.4**, we can build our Blending and Tuning Functions for the *LT* using in-parallel binary weight distribution for each algorithm.

Table 3.1- Tuning and Blending Function Typical Observation

Functions	Theoretical	Real/Experimental
$\mathbb{B}_{A_n}$	0.86	0.79
$\mathbb{T}_{A_n}$	0.93	0.87

Table 3.2- Observance of Error Functions in Typical Ratios for eABT

Random	err	Err
x (0.19,0.27,0.38)	0.0013	0.0004
y (0.27,0.39,0.64)	0.0008	0.0032
z (0.49,0.65,0.69)	0.0082	0.0193

3.2.3 Mathematical Models and Related Theory for eFES Unit Governance

Very similar to constructs we build for *eABT LT* module, this logical table also operates in the memory and is dynamically updated. It keeps tracks of the features $F(x, y, z) = \{F_1, F_2, \ldots\ldots\ldots, F_n, \}$ as *the ML* process evolves to accomplish the final optimum fitting after it has tried all the features from the pool. Similarly, it also stores data based on three dimensions, where *'x' = over-fitness, 'y' = under-fitness and 'z' = optimum-fitness.*

Features in the given datasets are of several types. They are also known as ‘attributes’ or ‘variables’. Type includes: *i)* numeric, such as continuous values such as time, speed, height and weight or discrete such as age, counts, and ii) categorical such as Gender, Color, Race, and Ranks. Some of the categories of features are linguistic, structural, and contextual.

Definition LT.5 - *Let there be two functions, Feature Adder as* $+\mathbb{F}$*, and Feature Remover as* $-\mathbb{F}$*, based on linearity of the classifier for each feature under test for which the RoOpF is valid (as described in Definition 3), and a feature is not repeated in the group.*

Construction - *eFES LT* module builds very important functions at initial layers for adding a good fit feature and removing a bad fit feature from the set of features available to it, especially when algorithm blend is being engineered. Clearly, as we discussed, not all features will have optimum predictive value and thus identifying them will count towards optimization. The feature adder function is built as:

$$+\mathbb{F}(x,y,z) \;=\; (F_n \cup F_{n+1}) \sum_{i=1}^{Z} (lt.score(i)) + \sum_{j,k=1}^{x,y} (lt.score(j,k) \tag{15}$$

The feature remover function is built as:

$$-\mathbb{F}(x,y,z) \;=\; (F_n \cap F_{n+1}) \sum_{j,k=1}^{x,y} (lt.score(j,k)) - \sum_{i=1}^{z} (lt.score(i)) \tag{16}$$

Very similar to k-means clustering[48] concept, that is highly used in unsupervised learning, *LT* implements feature weights mechanism(*FWM*) so it can report a feature with high relevancy score and non-redundant in a quantized form. Thus, we define:

$$FWM(X,Y,Z) = \sum_{x=1}^{X}\sum_{y=1}^{Y}\sum_{z=1}^{Z}(u_x w_x . u_y w_y . u_z w_z)(\Delta(x,y,z)) \tag{17}$$

$$\Delta(x,y,z) = \begin{cases} \prod_{l=1}^{L}(u_{lx}w_{lx}), & if\ z \neq 0, AND\ z > (0.5, y) \\ u_i \in \{0,1\},\ -1 \leq i \leq L & \\ \prod_{l=1}^{L}(u_{ly}w_{ly}), & if\ z \neq 0, AND\ z > (0.5, x) \end{cases} \tag{18}$$

Definition - LT.6 – *Let there be a Feature Scoring Function as FScore in LT module for which the correlation between each feature as accepted is minimum. Let Cor(x,y,z) be a function to compute the score for the feature sets as grouped in the LT object.*

Construction - *FScore(x,y,z)* and *Cor(x,y,z)* are functions on the second layer that ensure each entry is recorded in the *LT* object as the process continues. We build,

$$FScore(x,y,z) = \int_{D_i}(F_i|X,Y,Z)p_i dV_{x,y,z} \tag{19}$$

$$Cor(x,y,z) = \begin{cases} H(f_i) - H(f_i|f_{i+1}) \\ H(f_{i+1} - H(f_{i+1}|f_i) \\ H(f_i) + H(f_{i+1}) - H(f_i, f_{i+1}) \end{cases} \tag{20}$$

PROC LT.1

Import Features*:* a finite number of features Set F in n > 0, Integer T >0
Initialization: Define the categorical or numerical values, and set $F^{(n)} = \text{Constant value}$
For $F = \{F_1, F_2, F_3, \ldots\ldots, F_n,\}$
Select F(n) based on random function and define the distribution in space, $D[f(T)|F^{(n)} \in \partial F(F^{(n)})$
Update each $f \in F^{(n)}$, for which $f_n \geq F\{0.85,0:1\}$ is valid
Return f_x, f_y, f_z

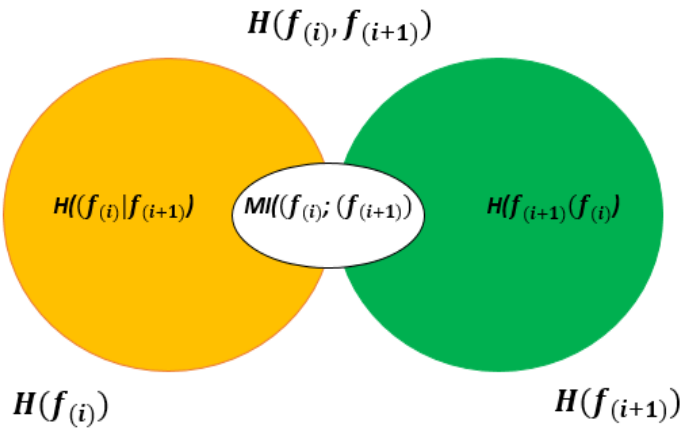


Figure 3. 5- Demonstration to illustrate the entropy-based feature distribution in space based on binary system.

Fig. 3.5 shows two colors (yellow and green) for same function related to each feature *(i)* and *(i+1)*. Thus, the entropy function is calculated in the inner layer of LT object as the features are added or removed based on matching scoring function.

Definition LT.7 – *Let lt.IrrF and lt.RedF be two functions to store the irrelevancy and redundancy score of each feature for a given data set in LT object and then correlates it for each test in blend of algorithms using lt.BlendAlgo Function, such that each feature obeys the condition* $0.3 > lt.BlendAlgo(lt.IrrF, lt.RedF) > 0.7\}$.

Construction - To construct *Irr.F* and *Red.F* , we implement Markov Blanket method in which we apply sequential filters to remove the feature one by one for higher *Red.F and Irr.F*. We alter the values between {-1 to +1) for theoretical consideration. It must be noted, that the values between {0 to 1} are realistic and mathematically possible. We build a mutual information (*MI*) function [100]so we can quantify the relevance of a feature upon other in the random set. This information is used to build the construct for *Irr.F*, as the classifier learns, it will mature the *Irr.F* learning module as defined in the algorithm *1*.

$$MI(Irr.F(x,y,z)|f_i,f_{i+1}) = \sum_{a=1}^{N}\sum_{b=1}^{N} p(f_i(a),f_{i+1}(b).log\left(\frac{f_i(a),f_{i+1}(b)}{p(f_i(a)\,.\,f_{i+1}(b)}\right) \quad (21)$$

$$Irr.F = \sum_{i,j}^{K}\begin{Bmatrix} f_{ii} & f_{ij} \\ f_{ji} & f_{ji} \end{Bmatrix} = = \begin{cases} MI(f_i;\ Irr.F) > 0.5 & Strong\ Relevant\ Feature \\ MI(f_i;\ Irr.F) < 0.5 & Weak\ Relevant\ Feature \\ MI(f_i;\ Irr.F) = 0.5 & Neutral\ Relevant\ Feature \end{cases} \quad (22)$$

We develop the relation of '*Irr.F'* and *MI* to show the irrelevancy factor and redundant factor based on binary correlation and conflict. Redundancy is another important quantity to compute for feature correlation, especially in classification problems. We use Markov Blanket [100][38].

Table 3.3- Tuning and Blending Function Typical Observation

Functions	Theoretical	Real/Experimental
$+\mathbb{F}$	0.89	0.81
$-\mathbb{F}$	0.94	0.93

Table 3.4- Observance of Error Functions in Typical Ratios for eFES

Random	err	Err
x (0.21,0.24,0.17)	0.0044	0.0017
y (0.31,0.49,0.79)	0.0008	0.0026
z (0.57,0.75,0.63)	0.0152	0.0057

Fig. 3.6 illustrates the *N*-experimental iteration of the conceptual flow shown. As we can observe, that *LT* governs the process at the lowest level. It creates the entry for each dimension *(X, Y, and Z).* As a threshold, if the *LT* value is less than 0.5, it is assigned binary '0', and if it is > 0.5, it assigns '1'. This shows the mechanics of the logical design of the algorithm being proposed. It shows that it may take *N* number of iterations to tune the table function. As discussed earlier, *LT* keeps track of the feature engineering for optimum fitting and outlier detection for a model being trained. Threshold is set to 50 % for *LT* function

return value. This shows that as features are added, the LT stays above 0.5, or features may need to be removed. The two-dimensional figures in **Fig. 3.6**(with blue and orange lines) demonstrate the underfitting and overfitting as the engine encounters and reports back to the LT object.

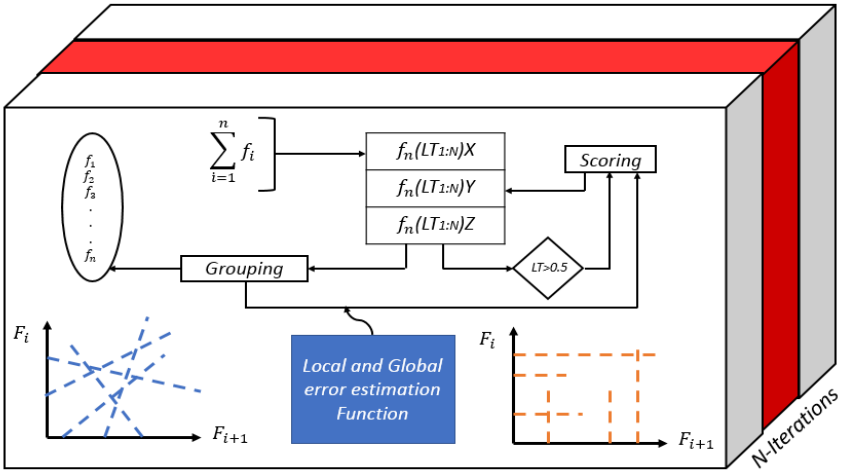


Figure 3. 6- Illustration of ***eFES*** Logical Table Internals

Finally, we build our cost and matrix function as follows:

$$Cost(LT, Elt) = \frac{1}{T} \sum_{x=1}^{X(t\in T)} \sum_{y=1}^{YX(t\in T)} \sum_{z=1}^{ZX(t\in T)} (Elt)_{x,y,z} \times M_x, M_y, M_z, \tag{23}$$

$$Where: M(x) = \begin{Bmatrix} x_{11} & \cdots & x_{1n} \\ \vdots & \ddots & \vdots \\ x_{n1} & \cdots & x_{nn} \end{Bmatrix}, M(y) = \begin{Bmatrix} y_{11} & \cdots & y_{1n} \\ \vdots & \ddots & \vdots \\ y_{n1} & \cdots & y_{nn} \end{Bmatrix}, M(z) = \begin{Bmatrix} z_{11} & \cdots & z_{1n} \\ \vdots & \ddots & \vdots \\ z_{n1} & \cdots & z_{nn} \end{Bmatrix}$$

3.3 Enhanced Algorithm Blend and Tuning(UNIT-A)

3.3.1 Introduction

The latest trend in the *ML* research[178] has shown an immense potential to evaluate numerous algorithms in parallel[41]. *eABT* unit implements built-in parallel processing by design. The following sections provide the theorems, mathematical internals, and the algorithms that collectively build the proposed *eABT* unit. This section provides the necessary details of *eABT* unit through the lens of underlying mathematical constructs, rules, procedures, algorithms, illustrations and framework. Equation (1) shows our master equation for *eABT* unit.

$$\left(eABT(A_i) = F_z + \sum_{A_n} (\upsilon_z)^{\beta-1} + \sum_{A_{n+1}} (\upsilon_z)^{\beta-2} +, \dots, \sum_{A_{n+m}} (\upsilon_z)^{\beta-m} \right) \tag{24}$$

In general computing, parallel processing is done by dividing program instructions to be run by multiple processors, so the time efficiency can be improved. This also ensures the maximum utilization of otherwise idle processors. Similar concepts can be implemented on the algorithms and *ML* models. *ML* algorithms depend on the problem and data types and require sequential training of each on the data models. However, the parallel processing can dramatically improve the learning process, especially for the blended engine, such as *eMLEE*. In the light of latest work of parallel processing in *ML*, such as: in [179], the authors introduced parallel framework on *ML* algorithms for large graphs. They experimented aggregation and sequential steps in their model to allow researchers to improve the usage of various algorithms. Another study done in [180], where authors used

induction to improve the parallelism in the decision trees. Authors in [74] introduced a python library Qjan to parallelize the *ML* algorithms in compliance by MapReduce. A PhD thesis[181] work done by a student @University of California at Berkeley, used concurrency control method to parallelize the *ML* process. Therefore, similar progresses have motivated us to incorporate parallel processing in the proposed work.

3.3.2 eABT with Bagging+ Conceptual Illustration

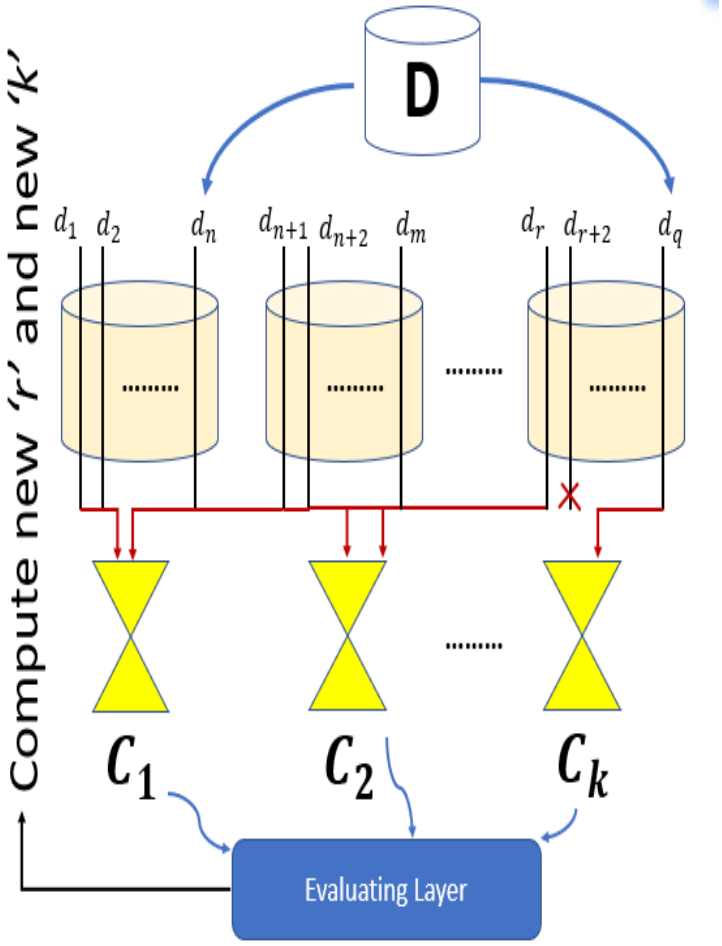


Figure 3. 7– Illustration of eABT influenced Bagging+ technique

This illustration shows the principle of eABT working when the Unit internal slices the data into various bags and then spread across various classifier for ensemble learning.

As it can be observed that random (stochastic) process are engineered to search for weak classifiers and then boost them based on split and measure approach.

3.3.3 Mathematical Constructs and Theoretical Groundwork

In this sub-section, we provide necessary definitions and the mathematical constructs .

Definition eABT.1 – *There must exist a matching factor (M.F) for optimum fitness between two methods being evaluated, so let* $\nabla d(A_1, A_2)$ *be the Euclidean distance between two methods. Let* $\mathbb{S}(x, y, z)$ *determines the suitability scores for the fitness of the given algorithm in 3D space. Let* $\mathfrak{C}$ *be an array that holds the OF and UF values for a given method.*

Construction – Here we create some vital functions to focus on fitness factors for building the higher layers of the Unit A.

$$\mathfrak{C}(x, y) = D(x, y) \left(\prod_{i(0), j(0)}^{I \times J} (w(A_{i,j}) \right) \tag{25}$$

$w_x \in \mathfrak{C}(x, y) \geq \left\| x^i + x^j \right\|$ *Signals very high OF and low bias*

$w_y \in \mathfrak{C}(x, y) \geq \left\| y^i + y^j \right\|$ *Signals very high UF and high bias*

$$\nabla d(A_1, A_2)_\mu = \sqrt{\left(\sum_{i=1}^{M} (\mu_i(p_i) - \left(\mu_i(p_i)\right)^2) \right)} \tag{26}$$

Let us assume a raw dataset to be $ds(\mathbb{s}, \mathbb{m})$ '$\mathbb{s}$' shows the signal and '$\mathbb{m}$*'* shows the noisy component of the dataset, and a cost classifier function $C(x, y, z)$, with a loss function as *L(x,y,z) | (0:1),* for which *n-sample* blocks are iterated such that loss function remains in the defined boundary as estimated, for which the feature sets exist in $F = \{f_1, f_2, f_3, \ldots \ldots, f_n\}$ with Matching Factor (*MF > 0.5)* as determined in the theorems. The classifiers of the *L* and *C* functions in the distribution '*D',* for *n* blocks of data sample, for the upper bounds of generalization error, is given by:

$$L(x,y,z) = \langle \mathbb{Q}\{(x_i, x_{i-1}), (y_i, y_{i-1}), z'\}\rangle \tag{27}$$

Where $\mathbb{Q}$,

$$\mathbb{Q}(f(x,y,z) = \sum_{j=1}^{N} \nabla d\left(A_j, A_{j+1}\right)_{\mu} \tag{28}$$

$$C(x,y,z) = \frac{1}{n \in N}\sum_{j=1}^{N} F_i\left(A_j, A_{j+1}\right)_{\mu} + \overline{\overline{\sqrt{err \in Errr^J}}} \tag{29}$$

Fig. 3.8 shows the spread of the real (red pentagons) and experimental (green diamonds) of the value of each *3D* coordinates when the blend is at random.

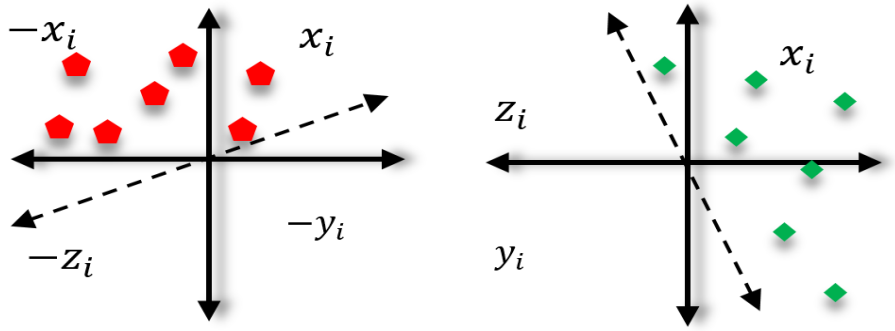


Figure 3. 8- Illustration of the concept of *3D* coordinates for each test algorithm.

RULE eABT.1

$$L(DS(\mathbb{s}(x,y,z), \mathbb{m}(x,y,z) = \begin{cases} 0, & (\mathbb{m}(x,y,z) \geq 0.5 \geq \mathbb{s}(x,y,z)) \\ 1, & (\mathbb{s}(x,y,z) \geq 0.5 \geq \mathbb{m}(x,y,z)) \end{cases}$$

Rule. eABT.1 estimates the loss function in signal data ($\mathbb{s}$) and noisy data ($\mathbb{m}$) that impacts the classifier design. Using *Novikoff's theorem*[177], for each algorithm in the pool, the *MF* function being quantized between {0:1} based on concept of *Lagragian distribution*[175].

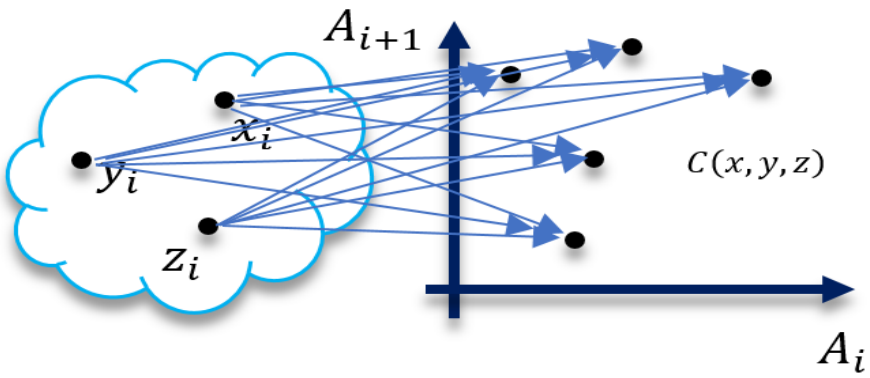


Figure 3. 9- Illustration of the cost function with logical coordinates in *3D* space.

As it can be observed that the cost function measures the algorithm fitness (between two adjacent algorithms while blend function is executed) in the optimized space.

Table 3.5 - Quantized Comparison of Cost Function

$C(x)$	$C(y)$	$C(x,y,z)$
0.84	0.64	0.36
0.79	0.93	0.57
0.73	0.77	0.28

Definition. eABT.2 - *Let $\psi(n)$ be the classifier function, that the model learns to be able to classify the optimum blend of algorithms for a given dataset and problem. Let $\mathscr{L}$ be a tuning parameter, on which the blending function as ϕ generates the correlating points for each dimension $\phi(x,y,z)$, such that the space is filled with many random points as* $\{\phi_1, \phi_2, \ldots\ldots, \phi_n\}$

Construction - As stated earlier, our engine is based on *3D space for x,y,z* dimensions that *eABT* unit uses to optimize the fitness of the blend to the given data. It must be noted that such approach is taken so the model gets very generalizable by design for any given data with any type of features. Thus, we manipulate matrix (real-valued) space to engineer the blend, thus, using *Frobenius norm*[175] form:

$$\|M\|(x,y,z) = \sqrt{\sum_{x=1}^{X}\sum_{y=1}^{Y}\sum_{z=1}^{Z} M_{x,y,z}^2} \tag{30}$$

Thus, using equations (42-46), and matrix manipulation for each dimension to evaluate the blend, we build:

$$\psi(n) = \underset{(x,y,z\,\in Z)}{Extract}(\,\|M\|(x,y,z)\,) \tag{31}$$

The relation between $\mathcal{L}$ and ϕ becomes extremely significant because if the training set is being picked by a random process, then the predictor selection also becomes a random process and a higher least square error is expected. Thus, the full certainty cannot be applied towards the classifier that will tune towards optimum fitness. This, we must construct a function that will interpret the success or failure rate of the classifier of the *eABT* towards achieving acceptable accuracy and will obey **Rule. eABT.1**. Recall that in *SL*, the end goal is always to label a function $f : x \rightarrow y$, that best fits the data (i.e., mapping function). Thus, we assume that the given data is split into two segments, (S_1 *and* S_2).

Lemma eABT.1 - $\psi(n) \leq \phi\,(z)$ *for all positive values of Op.F in the blend.*

If we use induction theory on each axis and crate a Boolean vector for z-dimension in which the matrix implication is all inclusive in each direction regardless of all possible extensions for z-value. Thus, we write three conditions:

a) $\phi_x \coloneqq \{(A_n \cup w(A_n)\} \rightarrow f(y,z) \approx err(0.8)$
b) $\phi_y \coloneqq \{(A_n \cup w(A_{n+1})\} \rightarrow f(x,z) \approx err(0.8)$
c) $\phi_z \coloneqq \{(A_n \cup w(A_{n+n})\} \rightarrow f(x,y) \approx err(0.8)$

By correlating *a), b), c),* we get

$$\Phi_{x.y.z} = \frac{1}{p}\sum_{j=1}^{N}\left(\prod_{i}^{N-1} w(A_n.A_{n_j}^{\ i}\right) \tag{32}$$

Definition. eABT.3 - *A GE (Err) is in bound 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* $err(A) \in Err(A+1), where\ 0.2 < e < 0.8$. *Let there be a maximum risk function* $(\mathcal{R})$ *with mean square error as* MSE *on the set of features as* $F = \{f_1, f_2, f_3, \ldots\ldots, f_n\}$.

Construction - We implement the maximum and minimum error bounds logical limits to ensure the optimum fitness (i.e. avoiding overfitting and underfitting) in-terms of the errors to be controlled by upper and lower bounds. We first force error to be at low threshold and then to be high threshold. Once the algorithm has learned the *max(e:0.8*) and *min(e:0.2*) bounds, it then learns to stay in between, and accuracy is maintained.

PROC eABT.1:

If (*LT.errFunc(0.8 > err*$\in Err >$ *0.2*) ***Then***

Record it and Send it to the Classifier Function

ElseIf (*LT.errFunc*(*err*$\in Err <$ *0.2*) ***Then***

Flag it to be '*O.F'*

ElseIf (*err*$\in Err >$ *0.8*) ***Then***

Flag it to be '*U.F'*

End if

Here we build an important function for optimization *GG* using *GF*. This function is based on important *ML* techniques and concepts known as *Orthonormalization*[182] and *Jacobian*[60].

$$GG(z) = GF \times \sum_{j\,=\,1}^{Nt}\left(\max(err)_j - \min(err)_j\right) \tag{33}$$

GF is the error gain factor and it is produced by the algorithms of the *eABT* unit. The errors produced by each algorithm, tend to increase when they are blended with each other, and errors of local and global functions must stay in the limit defined by **PROC eABT.1.** For general *ML* modeling, there are two categories of errors, *i) Estimation and ii) Approximation*. Collectively, we can call it generalization errors, in which our goal becomes a search for a special function $f'(x, y)$ that tends to minimize the risk of learning in the target space (*i.e.*, x ∈*X*, *y*∈*Y*, *z*∈*Z*), given by,

$$Risk[f']_{x,y,z} = \int_{X \times Y \times Z} L\,(y,\ f'(x,y))\, P(x,y,z)dxdydz \tag{34}$$

$P(x, y, z)$ will be unknown at this stage. We will have to approximate based on well-known mathematical and statistical learning theory[183], known as '*empirical risk minimization principle'*:

$$Risk_{emp}[f'] = \frac{1}{m}\sum_{i=1}^{m} L\,(y^i\ , f'\,(x^i)) \tag{35}$$

Here, we need to satisfy two conditions, as *i)* $\lim_{m \to \infty} Risk_{emp}[f'] = Risk[f']_{x,y,z}$ and *ii)* $\lim_{m \to \infty} \min_{f \in H} Risk_{emp}[f'] = \min_{f \in H} Risk[f]_{x,y,z}$. These two conditions will be valid when err is relatively small. The second condition requires minimal convergence.

The sub-estimator function is $\widehat{m_k} = c(F_k, \vartheta)$, Where ϑ is positive regularization parameters and it is observed that *c(f,0) = F* such that, $\vartheta = 0 \mid \widehat{m_\iota} = F_k$. We deduce that $\vartheta = \infty$, corresponds to maximal shrinking, that is $\{\widehat{m_k} = 0, for\ k = 1, \ldots, n\}$. Here, we can apply *Cross validation*[184] techniques (*CV)* and *Stein's unbiased risk estimate* (*SURE*), where popular estimators are (ridge), (lasso) and (pretest).

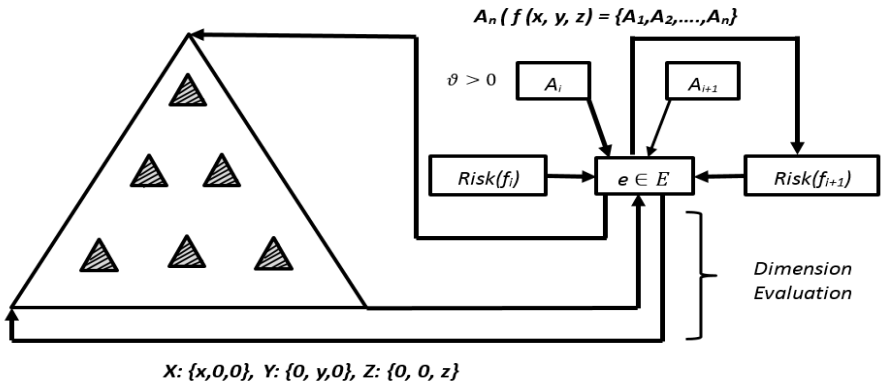


Figure 3. 10- Illustration of the risk estimation function in *3D* space for in-bound *LE and GE*.

The big and small triangles show the recursive process, where engine is taught to learn from its mistakes. It is conceptually shown with bounds of errors and risk functions so if the over-fitness is observed, the *LT* object is updated accordingly (governed by *eABT* algorithm 1 & 2).

PROC eABT.2

For *i* = 1 to *m* ***Do***
AA(i,j) ← *Risk* × $\prod_{k \leftarrow i,j} GF$
x+1 ← *X[i,i+1]*
y+1 ← *Y[i+1,i]*
z–1 ← *Z[i-1, y-1]*
End For

Definition eABT.4 - *Given a pool of supervised learning algorithm* $A(x, y, z) = \{A_1, A_2, \ldots\ldots\ldots, A_n\}$ *,that learns from the given data* $DS = \{ds_1, ds_1, ds_3, \ldots\ldots\ldots, ds_n\}$, *for which a correlation (matching)factor exists, such that* $MF(x \to 0, y \to 0, z \to \infty)$. *The local gain (LG) for each algorithm exists such that GG stays in optimized space given by* $g \in (1 - GG)\,||(A \in \{1,2,3,4, \ldots\ldots, n)\,||$.

Construction - Let us use the following three distance functions in the space, we use for inducing the optimum *GG*. Euclidean distance is widely implemented in *k-NN* algorithm,

$E.D = \sqrt{(x_1 - x_2)^2 + (y_1 - y_2)^2}$, and Similarity is based on *Hamming distance*, given by: $H.D = \sum_{j=1}^{k} |x_j - y_j|$, and *Minkowski distance* is defined as:

$$M.D = \left(\sum_{j=1}^{k} |x_j - y_j|^p \right)^{\frac{1}{p}} \tag{36}$$

Based on which, we construct the correlating matching factor *(MF*), at the point where each distance minimizes to its lowest possible value with minimum theoretical *GE* (*Err*). Thus:

$$MF(x, y, z) = \frac{1}{S} \sum_{i=1}^{n} (A_i) \pm \left\{ \frac{(E.D + H.D + M.D)}{n} \right\}, Err < 0.5 \tag{37}$$

We then construct a gain function for each algorithm, which depends on underlying performance metrics, such as accuracy, error, speed, complexity, overfitting, underfitting, and bias. For this layer, we consider only *Over-fitness (OF), Under-fitness (UF), and bias-ness (B)*, that affects the learning of the underlying algorithm. Thus, we can begin building gain function $GG = gain(A \in \{1,2,3,4,\ldots\ldots,n\})$. To develop the blend function, we formulate the *GG* such that *LG* for each blend minimizes the distance , so,$\Delta \leq |\{g \subseteq GG : g\ correlates\ GG\ for\ all\ feature\ sets.\}|$,

Lemma eABT.2 - $P(b_{old}), P(b_{new})$ *are two probability functions that represent the likelihood of algorithm fitness in the blend-based gain function.*

$$P(b_{new} = 1 \,|A_{(x,y,z)}) = \left(\frac{P(b_{new} = 1) P(b_{old} = 0}{P(b_{new})} \right) \tag{38}$$

Using *naïve bayes* approach,

$$P(x,y,z)_{new} = \prod_{j=1}^{N} P(x,y,z)_j^{new} \mid b_{new} \leq j \leq j+1 \tag{39}$$

If we set, $(x,y,z)_{new} \leftarrow (0,0,0)$, Then,

$$(x,y,z)_j^{new} \sim (x+1,y+1,z+1)_j^{new} \sim \ldots \sim (x+n,y+n,z+n)_j^{new} \tag{40}$$

Considering the fact, that *LG* will be very high in lower dimension will be very low in lower dimension, we tune the gain function such that in our final blend, we can filter the algorithm and progressive blend to be as optimum as possible. The approximation function (*AF*) correlates the scoring factor between set of predictor features, $S_{input} = \{1,2,3,4, \ldots \ldots, T_n\}$, and target variables, $S_{output} = \{1,2,3,4, \ldots \ldots, T_n\}$, as logically shown in algorithm definition. Thus, using equations (58) to (62), we construct:

$$GG = \times\ e^n \frac{1}{2} \sum_{i=1}^{m} \sum_{j=1}^{m} \alpha\beta\gamma_{i,j} \langle \alpha_{i,j}, \beta_{i,j}, \gamma_{i,j} \rangle + \sum_{k=1}^{K} ((\alpha\beta\gamma)_k) \tag{41}$$

Definition eABT.5 - *For each method 'A(i)', an approximation function exists between set of predictor features,* $S_{input} = \{1,2,3,4, \ldots \ldots, T_n\}$, *and target variables,* $S_{output} = \{1,2,3,4, \ldots \ldots, T_n\}$. *Each algorithm A in the blend performs well for all set of features F(n) included, where Weighted Performance Metric (eWPM)* ≥ 0.75) *for each metric stay above the threshold of measurement, till eWPM drops below 0.75 and* $A \in All\ in\ A_{x,y,z \leftarrow 0.1}$.

Construction - For the blend, we construct triangles in *3D* space using *axis align method*[13], Let $\Delta = \{\Delta_1, \Delta_2, \Delta_3, \ldots\ldots\ldots, \Delta_n\}$ indicates the infinite number of triangles in distributed space, among which , the following function defines the search for optimum coordinates:

$$\Delta_{(x,y,z)}^{(a,b)} = \{\mathbb{W}(x \leftarrow a, y \leftarrow b, z \leftarrow ab : a \leq b, and\ ab >\} \quad (42)$$

Where $\mathbb{W} = \begin{cases} 1, & a \leq b \\ 0, & ab > 0 \end{cases}$ and *eWPM* is available via *eMLEE API*.

For simplicity we assume, that *eWPM* will stay above 0.5 for optimum zone in our *3D* logical space. The co-ordinates in y-axis:

$Y_i = \alpha + \begin{bmatrix} \alpha \\ \beta_1 \\ \beta_n \end{bmatrix}_1 X_{i,1} + \ldots\ldots\ldots \begin{bmatrix} \alpha \\ \beta_1 \\ \beta_n \end{bmatrix}_n X_{i,n} + \epsilon_i$, Where $\epsilon = \begin{bmatrix} \epsilon_1 \\ \epsilon_2 \\ \cdots \\ \cdots \\ \epsilon_n \end{bmatrix}$ in each dimension where a blend of algorithms moves from higher dimension (*in x, y*) to a lower dimension for correspondence values of $\alpha\ and\ \gamma$. To validate and approve the algorithm blend after tuning has been achieved the error that appears as a complex function, must be observed depending on many factors and underlying blend of algorithms. *LE* exhibited by each algorithm is no more valid in its mathematical form, thus we develop a complex error function. The Errors produced by each algorithm, tend to increase when they are blended with each other. A popular statistical technique that accomplishes the modeling of relationship between *n+* variables, based on linear equation for its fitness is built.

$$\sigma = \sqrt{\left(\frac{\sum_j {\gamma^2}_j}{(n-p-1)}\right)} \quad (43)$$

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

Using *Bernoulli Distribution*, we can write the distribution of Y_i: $Prob\{Y_i = y_i\} = \pi_i^{y_i}(1 - \pi_i)^{1-y_i}$. Thus, we formulate two very important constructs for tuning the engine internal layer to the limits imposed by Rule 3**.** Hereby, we apply *stacked generalization technique* and *cascading method*[18] so *LE* in out of bound (*err > 0.8 OR*

err< 0.2) and LE in bound (*err> 0.2 AND err < 0.8*) confidence score is close to the highest posterior of the learner, and thus it will improve the rejection of learner pointers that may overfit or underfit the model. The following conditions are preliminaries for the formulation of these bounds. $\widehat{x_t} := w + x_t y - y_t x$, $y_i \leftarrow x + y_t$, $\widehat{y_t} := max\{< w, err(z) > err(x,y)$. Thus,

$$\left\|\begin{bmatrix} x_i \\ \vdots \\ x_n \end{bmatrix}\right\| + \left\|\begin{bmatrix} y_i \\ \vdots \\ y_n \end{bmatrix}\right\| \leq \sqrt{\langle \begin{bmatrix} \sigma(x_i) \\ \vdots \\ \sigma(x_n) \end{bmatrix}, \begin{bmatrix} \sigma(y_i) \\ \vdots \\ \sigma(y_i) \end{bmatrix}, \ldots\ldots, \begin{bmatrix} \sigma(x_{i+m}) \\ \vdots \\ \sigma(x_{n+m}) \end{bmatrix} \rangle}$$

By conjecturing above conditional limits, we get:

$$LE(OOB) = \prod_{i=1}^{N} \left(\widehat{x_{t(-\iota)}} - \widehat{y_{t(-\iota)}}\right) \tag{44}$$

$$LE(IB) = \prod_{i=1}^{N} \left(\widehat{x_{t(+\iota)}} + \widehat{y_{t(+\iota)}}\right) \tag{45}$$

Lemma eABT.3 - *$eWPM = \mathbb{W} > 0.75$ for all the sets, in which the following defines the boundary as,* $(GG/(g+1) + (\sigma)^3$

Using *Slutsky's theorem and Delta method*[177], we state that two sets of random algorithms, as rA_i, rA_{i+1}, and a probability *Pr* shows the relation exists so the continuous function *c(x,y)* converges for the random algorithms pick into *c*. Thus, we generate the following conditions that are necessary for the blending function to maintain the *eWPM* boundary conditions.

i) LG and *GG* correlate in lower dimensions if $\Pr(rA_i) > 0.5$ for each occurrence of $g(x,y,z) \cong A\left(\Delta_{(x,y,z)}^{(a,b)}\right)$, and, *ii)* when *eWPM* stays under 50 % or 0.5, then $Op.F \cong$

$(R(UF) \gg 0.8(err)$. As governed by the Algorithm 1 and 2, the AF function finally follows the construct as we build,

$$AF = A^{z \in Z} \times \left[\prod_{i,j,k}^{N \times M} \Delta_{(x,y,z)}^{(a,b)} \right] Prob\{Y_i = y_i\} \tag{46}$$

Using *Induction theory* and *Lagrangian distribution*, we set the following terms, so *eWPM* can regulate the blended model classifier learning in the bounds as stated earlier.

$$= mean\Delta(\langle x_i, y_i \rangle) + n^2 + 2x_i y_i \|x_{i+n} y_{i+n}\|$$
$$\geq max\Delta(\langle x_{i+1}, y_{i+1} \rangle) + (n-1)^2 + 2x_i y_i \|y_{i+n}\| + \|x_{i+n}\|^2$$
$$\leq min\Delta(\langle x_{i+n}, y_{i+1} \rangle) + (n-2)^2 + 2x_i y_i \|x_{i+n}\| + \|y_{i+n}\|^2$$

By conjecturing these, we get

$$\mathbb{W}(z, \sigma) = \frac{1}{2} \sum_{i}^{x} \sum_{j}^{x,y} \sum_{k}^{x,y,z} \sigma_{ijk} \|z_k - xj_k\|^2, AF(-1, +1) \tag{47}$$

PROC eABT.3

Initialize *x,y and z*

While *LG and GG in (x,y) > f(z) <= 0* ***do***

f (z∈ f(x, y, z) ← LG(z) + ($\|\boldsymbol{z_k - xj_k}\|^2$*)*

Input algorithms with positive *AF*

End While

Update *z+i*

output *GG(z)*

Definition eABT.6 - *ζ(tp,fp,tn,fn) is a function that computes the Unit's ability to learn from mistakes(LFM) and measures the GG such that* $\zeta \cong \nabla\left(\sqrt[3]{(GG+1)/GE^2}\right)$*, and the accuracy of the classifier remains traversal distributed in the lower dimension of z with*

minimal risk function spread evenly through predicted probability as a function of, and greater than $\frac{1}{2}\ |Pr(z)|$

Construction - For each entry in the *LT* object, there is a function that second layer of the unit creates, and then using the earlier hypothesis, we feed the predictions that has high *GE* and low *GG*, with absolute values less for the matrices averages. We create a general learning rule for mistakes-based training, in which the following can be assumed. *a)* $ds(LE: -1: +1) \leftarrow LG \in GG(z), b)\, CF(z) \leq 0.8(GE) + max(LF(x, y), and\, c)\, \Delta\mathbb{E} \leq log(\partial z) \neq 0.$ ∂z represents the partial changes in *z*. $\Delta\mathbb{E}$ indicates the rate of change of *GE* when the internal layer is fed back with the errors. We construct the following classes on which the *eABT* unit develops unique learning fashion purely based on mistakes it has made in the last testing phase. Thus, for next training phase the prediction scores are stored in *LT* object and weights are distributed evenly across each axis, as illustrated in **Fig. 3.11**

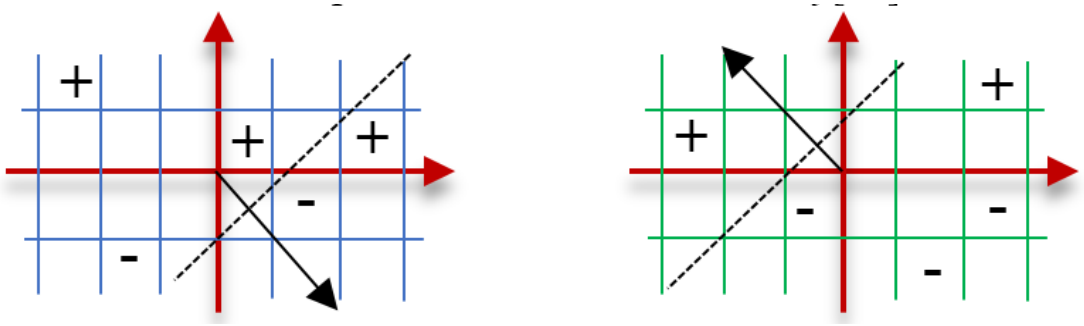


Figure 3. 11- Illustration of binary weighted vectors for LFM Module.

Fig. 3.11 shows classification techniques to follow the vector orientation at any given point in time during classifier learning when the data segments (i.e., array by array) are fed back regulated by LT objects, as defined in Algorithm 4. Using Jacobian and Laplace kernel methods, we can construct the function, as

$$\zeta(tp, fp, tn, fn) \;= \frac{1}{m}\left(\sum_{i=1}^{m} M_{LE}\sqrt{(y - y_i) \;+\; (x - x_i)/(y)}\right) \tag{48}$$

M_{LE} is based on logistic function for the classification of the sub-learners in which the *FF* quantification is observed at least above 50 %. We consider such feedback to be useful for training purposes. Other than this, we suspect outliers and disregard from our training set, therefore, we can construct two internal functions of *LFM* layer, as

$$\langle \mathbb{H}(\boldsymbol{x}, \boldsymbol{y}), \mathbb{H}(\boldsymbol{z})\rangle = \sum_{\boldsymbol{i_1}} \dots \sum_{\boldsymbol{i_2}} \|\zeta_{i_1}\| \times \|\zeta_{i_2}\| \times \dots \times \boldsymbol{k}\|\zeta_{i_n}\| \;= \left(\sum_{k} \zeta_{k_n}\right)^{k\in K+1} \tag{49}$$

Thus, for all symmetrical kernel that are based on semi-definite functions of *LFM* layer, we can use concepts of nonuniform learnability[13], we write finally,

$$\langle \mathbb{Z}(x, y), \mathbb{Z}(z){:}\,1\rangle \;=\; \langle \sum_{n\in N} \zeta_{z_n} + \sum_{n\in N} \zeta_{x_n} \,, \sum_{n\in N} \zeta_{y_n}\rangle = 1 \tag{50}$$

$$\langle \mathbb{Z}(x, y), \mathbb{Z}(z){:}\,0\rangle \;=\; \langle \sum_{n\in N} \zeta_{z_n} - \sum_{n\in N} \zeta_{x_n} \,, \sum_{n\in N} \zeta_{y_n}\rangle \;= 0 \tag{51}$$

Table 3.6 - Conditions for LFM based on Sauer's Lemma and Growth Function[13]

Conditions	**Definitions**
$GG \;\geq\; \|\mathbb{W}^n - 1\|$	*Inequality gain constraint*
$GE \leq \left\|(x, y, z)_j^{new}\right\|^2$	*Error monotonicity limit*
$Risk_{emp}[f'] \;\leftarrow\; GE(z)$	*Risk verification*
$\phi_{x.y.z} \;\leftarrow Pr\big(gg(z)\big) = 1$	*Probability Filtering*
$x_i \cong\; y_i \;\rightarrow\; y_i \cong x_i$	*Symmetrical Reflex*
$\alpha\beta\gamma(\zeta) \;\rightarrow\; [\sigma(x_i, (y_i), (z_i))]$	*Fitness Correlation*

3.4 Enhanced Feature Engineering and Selection(UNIT-B)

3.4.1 Introduction

In this section, we detailed the modeling of eFES unit with specially designed constructs that aims to aggregate the right feature set with quantification and predictive value computation done by the internal Unit and LT.

3.4.2 Mathematical Constructs and Theoretical Groundwork

Definition eFES.1. *Let there be a Logical Table (LT) module that regulates the ML process during eFES constructions. Let LT have 3D coordinates as x, y, and z to track, parallelize, and update the* $x \leftarrow overfit(0:1), y \leftarrow underfit(0:1), z \leftarrow optimumfit\ (-1:+1)$. *Let there be two functions, Feature Adder as* $+\mathbb{F}$, *and Feature Remover as* $-\mathbb{F}$, *based on linearity of the classifier for each feature under test for which the RoOpF (Rule. 1) is valid. Let Lt. RoOpF > 0.5 to be considered of acceptable predictive value.*

Construction - eFES LT module builds very important functions at initial layers for adding a good fit feature and removing a bad fit feature from the set of features available to it, especially when algorithm blend is being engineered. Clearly, not all features will have an optimum predictive value and thus identifying them will count towards optimization. The feature adder function is built as:

$$+\mathbb{F}(x,y,z) = +\mathbb{F}_{F_n} = \ (F_n \cup F_{n+1}) \sum_{i=1}^{Z} (LT.score\ (i)) + \sum_{j,k=1}^{x,y} (LT.score\ (j,k)) \tag{52}$$

The feature remover function is built as:

$$-\mathbb{F}(x,y,z) = -\mathbb{F}_{F_n} = \ (F_n \cap \ F_{n+1}) \sum_{j,k=1}^{x,y} (LT.score\ (j,k)) - \sum_{i=1}^{z} (LT.score(i)) \tag{53}$$

Very similar to *k*-means clustering [12] concept, that is highly used in unsupervised learning, LT implements feature weights mechanism (FWM) so it can report a feature with high relevancy score and non-redundant in a quantized form. Thus, we define:

$$FWM(X,Y,Z) = \sum_{x=1}^{X}\sum_{y=1}^{Y}\sum_{z=1}^{Z} (u_x w_x .\ u_y w_y .u_z w_z)\ (\Delta\ (x,y,z)) \tag{54}$$

$$\Delta(x,y,z) = \begin{cases} \prod_{l=1}^{L} (u_{lx} w_{lx}), & \text{if } z \neq 0, AND\ z > (0.5, y) \\ u_i \in \{0,1\}, \ -1 \leq i \leq L & \\ \prod_{l=1}^{L} (u_{ly} w_{ly}), & \text{if } z \neq 0, AND\ z > (0.5, x) \end{cases} \tag{55}$$

It shows that it is based on binary weighted classification scheme to identify the algorithm for blending and then assign a binary weight accordingly in LT logical blocks. The diamond shape shows the err distribution that is observed and recorded by LT module as new algorithm is added or existing is removed. We finally provide the eFES LT functions as:

$$eFES^{\boxplus} = \ [\mathbb{R}_{eFES} \ = \frac{1}{N_e}\left(\sqrt{\frac{err}{err \ + \ Err}}\right)^2] \ \times \sum_{n=1}^{N} F_n(f(x,y,z) \left| exp\left(\frac{+\mathbb{F}_{F_n}}{+\mathbb{F}_{F_n} + \ (-\mathbb{F}_{F_n})}\right)\right| \tag{56}$$

where *err* = local error (LE), *Err* = global error (GE). *f (x,y,z)* is the main feature set in 'F' for 3D.

Definition eFES.2 - $F_n = \{F_1, F_2, F_3, \ldots\ldots, F_n\}$ *indicates all the features appears in the dataset, where each feature* $F_i \in F_n \mid f_w \geq 0$. f_w *indicates the weighted feature value in the set. Let* $F_{ran}\ (x, y, z)$ *indicates the randomized feature set.*

Construction - We estimate the cost function based on randomized functions. Las Vegas and Monte Carlo algorithms are popular randomized algorithms. The key feature of the Las Vegas algorithm is that it will eventually have to make the right solution. The process involved is stochastic (i.e., not deterministic) and thus guarantee the outcome. In case of selecting a function, this means the algorithm must produce the smallest subset of optimized functions based on some criteria, such as the accuracy of the classification. Las Vegas Filter (LVS) is widely used to achieve this step. Here we set a criterion in which we expect each feature at random gets a random maximum predictive value in each run. $\emptyset$ shows the maximum inconsistency allowed per experiment.

PROC eFES.1

$Score_{best} \leftarrow Import\ all\ attributes\ as\ 'n'$
$Cost_{best} \leftarrow n$
For $j \leftarrow 1\ to\ Iteration_{max}$ ***Do***
    Cost ← Generate random number between 0 and $Cost_{best}$
    Score ← Randomly select item from Cost feature
    If LT.InConsistance ($Score_{best}$, Training Set) $\leq \emptyset$ ***Then***
        $Score_{best} \leftarrow Score$
        $Cost_{best} \leftarrow C$
    End If
End For
Return ($Score_{best}$)

Definition eFES.3 - *Let lt.IrrF and lt.RedF be two functions to store the irrelevancy and redundancy score of each feature for a given dataset.*

Construction - Let us define a Continuous Random Vector $CRV \in Q^N$, and Discrete Random Variable $DRV \in H = \{h_1, h_2, h_3, \ldots\ldots, h_n\}$. The density function of the random vector based on cumulative probability is $P(CRV) = \sum_{i=1}^{N} P_H(h_i) p\ CRV \mid DRV$, $P_H(h_i)$ being a priori probability of class.

Overlapped matrices for *Red.F* and *Irr.F*, for which the probability scope resulted in acceptable errors by stepping into vector space, for which 0.8 > err > 0.2. As we observe that the higher error limit (e) (err, green line, round symbol) and lower error limit (E), (Err, blue line, square symbol) bound the feature correlation in this process. Our aim is to spread the distribution in z-dimension for optimum fitting as features are added. The red line (diamond symbol) that separates the binary distribution of Redundant Feature (Red.F) and Irrelevant Features (Irr.F) based on error bounds. The green and red lines define the upper and lower limit of the error, in which all features correlate. Here, we build a mutual information (MI) function [100] so we can quantify the relevance of a feature upon other in the random set and this information is used to build the construct for *Irr.F*, since once our classifier learns, it will mature the *Irr.F* learning module as defined in the algorithms later in the section.

$$MI(Irr.F(x,y,z)|f_i, f_{i+1} = \sum_{a=1}^{N}\sum_{b=1}^{N} p\,(f_i(a), f_{i+1}(b).log\left(\frac{f_i(a), f_{i+1}(b)}{p\,(f_i(a)\,.\,f_{i+1}(b)}\right) \quad (57)$$

We expect $MI \leftarrow 0$, for features to be statistically independent, so we build the construct in which the MI will be linearly related to the entropies of the features under test for *Irr.F* and *Red.F*, thus:

$$M.I(f_i, f_{i+1}) = \begin{cases} H(f_i) - H\,(f_i|f_{i+1}) \\ H\,(f_{i+1} - H\,(f_{i+1}|f_i) \\ H(f_i) + H(f_{i+1}) - H\,(f_i, f_{i+1}) \end{cases} \quad (58)$$

We use the following construct to develop the relation of '*Irr.F'* and '*Red.F'* to show the irrelevancy factor and Redundant factor based on binary correlation and conflict mechanism.

$$Irr.F = \sum_{i,j}^{K} \begin{Bmatrix} f_{ii} & f_{ij} \\ f_{ji} & f_{ji} \end{Bmatrix} Red.F \quad (59)$$

$$= \begin{cases} MI(f_i;\, Irr.F) > 0.5 & Strong\ Relevant\ Feature \\ MI(f_i;\, Irr.F) < 0.5 & Weak\ Relevant\ Feature \\ MI(f_i;\, Irr.F) = 0.5 & Neutral\ Relevant\ Feature \end{cases}$$

Definition eFES.4 - *Globally in 3D space, there exist three types of features types (variables), as predictor features:* $PF = \{pf_1, pf_2, pf_3, \ldots\ldots pf_n\}$*, and accepted features to be* $AF = \{af_1, af_2, af_3, \ldots\ldots, af_n\}$ *and rejected features to be* $RF = \{rf_1, rf_2, rf_3, \ldots\ldots rf_n\}$*, in which* $\mathbb{G} \geq (g+1)$*, global gain for all experimental occurrence of data samples.* $'\mathbb{G}'$ *being the global gain (GG).* $'g'$ *being the local gain (LG). Let PV be the predictive value. Accepted features are* $af_n \in PV$*, strongly relevant to the sample data set* ΔS*, if there exist at-least one x and z or y and z plane with score* ≥ 0.8*, AND a single feature* $f \in F$ *is strongly relevant to the objective Function 'ObF' in distribution 'd' if there exist at-least a pair of example in data set* $\{\Delta S_1, \Delta S_2, \Delta S_3, \ldots\ldots, \Delta S_n \in I\}$*, such that* $d(\Delta S_i) \neq 0$ *and* $d(\Delta S_{i+1}) \neq 0$*. Let* $\nabla(\varphi, \rho, \omega)$ *correspond to the acceptable maximum 3-axis function for possible optimum values of x, y, and z respectively.*

Construction - We need to build an ideal classifier that learns from data during training and estimate the predictive accuracy, so it generalizes well on the testing data. We can use probabilistic theory of Bayesian [31] to develop a construct similar to direct table lookup. We assume a random variable to be 'rV' that will appear with many values in set of $\{rV_1, rV_2, rV_3, \ldots, rV_n\}$ that appear as a class. We will use prior probability $P(rV_i)$. Thus,

we represent a class or set of classes as rV_i, and the greatest $P\ (rV_i)$, for given pattern of evidence (*pE)* that classifier learns on $P\ (rV_i \mid pE) > P\ (rV_j \mid pE)$ *valid for all* $i \neq j$, Because we know that

$$P\ (rV_i \mid pE) = \frac{P\ (pE \mid rV_i)\ P\ (rV_i)}{(P(pE))} \tag{60}$$

Therefore, we can write the conditional equation where P (*pE*) is considered regarding probability of (*pE*) is $P\ (pE \mid rV_i)P\ (rV_i) > P\ (pE \mid rV_j)P\ (rV_j)$ *valid for all* $i \neq j$. Finally, we can write the probability of the error for the above given pattern, as *P* (*pE*)|*error*, assuming the cost function for all correct classification is 0, and for all incorrect is 1, then as stated earlier, the Bayesian classification will put the instance in the class labelling the highest posterior probability as $P\ (pE) = \sum_{i=1}^{k} P\ (rV_i)\ P\ (pE|rV_i)$. Therefore, the construct can thus be determined as $P\ (pE)|error = Error\ [1 - max\{P\ (rV_1) \mid pE, \ldots\ldots, P\ (rV_k|pE)\}]$. Let us construct the matrix function of all features, accepted and rejected features, based on GG and LG, as

$$\mathbb{G}\ (x, y, z) = \frac{1}{N} \sum_{i=1}^{n} \{(g_i) \times MH\} \tag{61}$$

$$MH = \begin{Bmatrix} pf_{x1y1} & \cdots & pf_{x1yn} \\ \vdots & \ddots & \vdots \\ pf_{xny1} & \cdots & pf_{xnyn} \end{Bmatrix} = \begin{Bmatrix} af_{11} & af_{12} & \ldots & af_{1n} \\ af_{n1} & af_{n2} & \ldots & af_{nm} \end{Bmatrix} \times \begin{Bmatrix} rf_{11} & rf_{1n} \\ rf_{21} & rf_{2n} \\ rf_{2n} & rf_{mn} \end{Bmatrix} \pm \nabla\ (\varphi, \rho, \omega) \tag{62}$$

Using Naïve Bayes multicategory equality as:

$$P_{1,2,3,\ldots,N}\left[\sum_j x_j\right]+\left[\sum_j y_j\right]+\left[\sum_j z_j\right]=\sum_k Var(x,y,z)[z^{*i}] \tag{63}$$

where $z^*(n) \coloneqq \underset{z}{\operatorname{argmax}} P(z)\prod_{k=1}^{n} p([z]).z_k$, and Fisher score algorithm[25] can be used in FS to measure the relevance of each feature based on Laplacian score, such that $B(i,j)=\begin{cases}\frac{1}{N_l}\ if\ u_i=u_i=1\\ 0\ otherwise,\end{cases}$., N_l shows the no. of data samples in test class shown subscript 'l'.

To group the features based on relevancy score, we must ensure that each group member of the features exhibit low variance, medium stability and their score is based on optimum-fitness, thus each member follows $k\ (k \in K, where\ K \leq f\ (0{:}1)$. This also ensure that we address the high dimensionality issue, as when feature appears in high dimension, they tend to change their value for training mode, thus, we determine the information gain using entropy function as:

$$Entropy\ (F_n) = \sum_{t=1}^{V1} -p_t\ log\ p_t \tag{64}$$

RULE eFES.1

If (g (err) < 0.2) ***Then***

Flag 'O.F'

Elseif (*g* (*err*) > 0.8**)** *Flag 'U.F'*

If we assume the fact of $\{\Delta S_1, \Delta S_2, \Delta S_3, \ldots\ldots, \Delta S_n \in I\}$, such that $d\ (\Delta S_i) \neq 0$ and $d\ (\Delta S_{i+1}) \neq 0$, where '$I$' is the global input of testing data. We also confirm the relevance of the feature in the set using objective Function construct in distribution 'd', thus:

$$ObF\ (d,I)\ = \frac{log\ (Gain\ (I,F_{(t:x,y,z)}))}{(err[max:1],err[min:0])}\ \ |\ d\ (\Delta S_i)\ \neq\ 0\ |\ \ for\ every\ F_i\ in\ group \tag{65}$$

Then, Using Equations (14)–(17), we can finally get

$$F.Eng(x,y,z) = \frac{1}{(k \times M)} \sum_{t=1}^{K} \prod_{t=k}^{M} ObF\ (d,I)\ \times MH_t) \tag{66}$$

$$F.Grp(x,y,z) = F.Eng(x,0,0) + F.Eng(0,y,0) - F.Eng(0,0,z) \tag{67}$$

Fig. 3.12 shows the Illustration of Feature Engineering and Feature Group as constructed in the mathematical model and governed by the Algorithms 8, defined later. Metrics API is available from *eMLEE* package. The white, yellow, and red orbital shapes indicate the local gain progression through 3D space. The little 3D shapes (*x*, *y*, and *z*) in the accepted feature space in grouping indicates several (theoretically unlimited) instances of the optimized values as the quantization progresses.

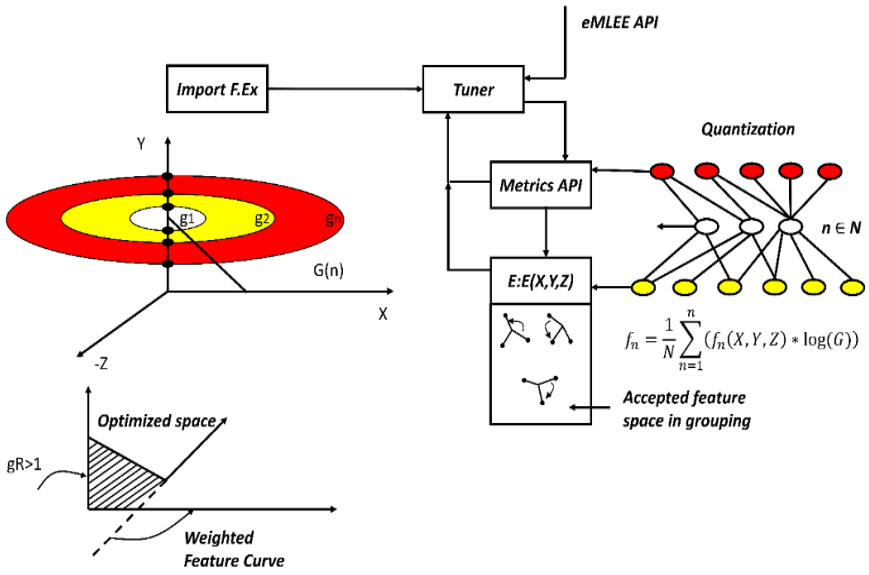


Figure 3. 12- Illustration of Feature Engineering and Feature Group as constructed in the mathematical model.

Definition. eFES.5 - *Feature selection is governed by satisfying the scoring function (score) in 3D space (x:Over-Fitness, y:Under-Fitness, z:Optimum-Fitness) for which evaluation criterion needs to be maximized, such that* $Evaluation\ Criterion: f'$. *There exist a weighted,* $W(\emptyset)\{\nabla(\varphi, \rho, \omega), 1\}$ *function that quantifies the score for each feature, based on response from eMLEE engine with function* $eMLEE_{return}$, *such that each feature in* $\{f_1, f_2, f_3, \dots\dots\dots, f_n, \}$, *has associated score for* $(\varphi: x, y, z, \rho: x, y, z, \omega: x, y, z)$.

Two or more features may have the same predictive value and will be considered redundant. The non-linear relationship exists between two or more features (variables) that affects the stability and linearity of the learning process. If the incremental accuracy is improved, then non-linearity of a variable is ignored. As the number of the features are added or removed in the given set, the OF, UF, and B changes. Thus, we need to quantify their convergence, relevance, and covariance distribution across the space in 3D. We implement weighted function for each metric using LVQ technique [18], in which, we measure each metric over several experimental runs for enhanced feature set, as reported back from the function explained earlier, such that we optimize the z-dimension for optimum fitness and reduce *x* and *y* dimension for over-fitness and under-fitness. Let us define:

$$W(\emptyset) = \frac{1}{\int_{S_t} p(x)dx} \sum_{\gamma=1}^{\sigma} N_\gamma^T . N_\gamma \int_{S_\gamma} p(x)dx \tag{68}$$

where the piecewise effective decision border is $S_t = \sum_{\gamma=1}^{\sigma} S_\gamma$, In addition, the unit normal vector, (N_γ) for border $S_\gamma, \gamma = 1, 2, 3, 4, \dots\dots\dots \sigma$ is valid for all cases in space.

We used the Las Vegas Algorithm approach that helps to get correct solution at the end. We used it to validate the correctness of our gain function. This algorithm guarantees correct outcome if the solution is returned or created. It uses the probability approximate functions to implement runnable time-based instances. For our feature selection problem, we will have a set of features that will guarantee the optimum minimum set of features for acceptable classification accuracy. We use linear regression to compute the value of features to detect the non-linearity relationship between features, we thus implement a function, $Func\big(U(t)\big) = a + b * t$. Where a, and b are two test features and values can be determined by using linear regression techniques, so $b = \frac{\sum_{t=1}^{T}(t - \overline{t})\,(U(t) - \overline{u})}{\sum_{t=1}^{T}\,(t - \bar{t})^2}$, Where,$a = \bar{u} - b * \bar{t}$,$\bar{u} = \frac{1}{T}\sum_{t=1}^{T} U\,(t)$, $\bar{t} = \frac{1}{T}\sum_{t=1}^{T} t$. These equations also minimize the squared error. To compute weighted function, we use feature ranking technique [48]. In this method, we will score each feature, based on quality measure such as information gain. Eventually, the large feature set will be reduced to a small feature set that is usable. The Feature Selection can be enhanced in several ways such as pre-processing, calculating information gain, error estimation, redundant feature or terms removal, and determining outlier's quantification, etc. The information gain can be determined as:

$$Gain_I(w) = -\sum_{j=1}^{M} P\big(M_j\big).\log P\big(M_j\big) + P(w)\sum_{j=1}^{M} P\,(M_j \mid w).\log P\big(M_j \,\big|w\big) + P\,(\overline{w})\sum_{j=1}^{M} P\big(M_j \,\big|\, \overline{w}\big)\,.\log\, P\,(M_j|\overline{w}) \tag{69}$$

'*M*' shows the number of classes and '*P'* is the probability. '*W*' is the term that it contains as a feature. $P\,\big(M_j\,\big|w\big)$ is the conditional probability. In practice, the gain is normalized using Entropy, such as

$$Norm.Gain_I(w) = \frac{\{Gain_I(w)\}}{\left\{-\frac{n(w)}{n}\log\frac{n\,(w)}{n}\right\}} \tag{70}$$

Here we apply conventional variance-mean techniques. We can assume, $\max \nabla \sum_{i=1}^{n} \varphi_i \rho_i \omega_i - \sum_{i=1}^{n} \log \varphi_i \rho_i \omega_i$. The algorithm will ensure that 'EC{F. Sco (x, y, z), F. Opt (x, y, z) $\geq$ 0.5}' stays in optimum bounds. Linear combination of Shannon information terms [60] and conditional mutual information maximization (*CMIM*) [25] for $U_{MAX}(Z_k) = \max_{Z_k \in \Delta s}[Inf(Z_k: X, Y|(XY)_k)]$ builds the functions as

$$Score(X|Y) = \sum_{y_k \in Y} G(y_k). \sum_{x_{k\prime} \in X} G\,(x_{k\prime}) \times \log\,(g(z)) \tag{71}$$

$$J_{MIN}(Z)^d = -\beta\,(\prod_{k,k\prime}^{K\,(0)} S(X:Y)_k + \gamma\,(\prod_{k,k\prime}^{K\,(0)} S(Y:X)_{k\prime} \tag{72}$$

By using Equations (69)– (72), we get

$$F.Sco(x,y,z) = Score(X|Y) + \sum_{i=1}^{n} W(\emptyset)_i - \sum_{j=1}^{n} Gain_j(w) \tag{73}$$

$$F.Opt(x,y,z) = J_{MIN}(Z)^d. \prod_{F.Soc(x,y,z)}^{N} \left\{\frac{F.Soc(x,y,z)}{1 + \mathrm{T}Norm.Gain_I(w)}\right\} - \sum_{j=1}^{n} \Delta Err\,(j) \tag{74}$$

3.5 eMLEE Engine Structure

3.5.1 Introduction

eMLEE reveals itself to the outer world as an enhancement to the ensemble approach based on mathematical and stochastic thinking. It acts a general engine housing LT, eABT, and eFES units in its technical layers. It further provides various options to be used for predictive machine learning model training. As shown in **Fig 3.13- 3.17**, it constructively improves the standard classifier learning process using internal units. Thus, to fully utilize its potential, it provides options to tune the parameters of the engine as desired by the end goal. For example, if features come in a small set or feature engineering is not desired or priority, the users can option out and only focus on ensemble part of the engine. Similarly, users can customize the metrics-trade off on demand using APIs and functions calls.

3.5.2 eMLEE System Conceptual Illustrations

In this section we provide some useful illustrations to understand the construction of the eMLEE as an engine.

Fig. 3.13 shows the parallel processing of the engine. *dq* shows the raw data, *An* shows the methods pool, *Bag+* shows the enhanced bagging technique, and *Bl+* shows the enhanced blending technique. Parallel processing is achieved implicitly by design. While the bags are created for classifier(s) to train on, the *LT* object (in parallel) records the

learning metrics in its table based on *eWPM* and decide the weights of each classifier and feature and once some threshold is met, it eventually decides the maximum learning performance has met for any number of iterations.

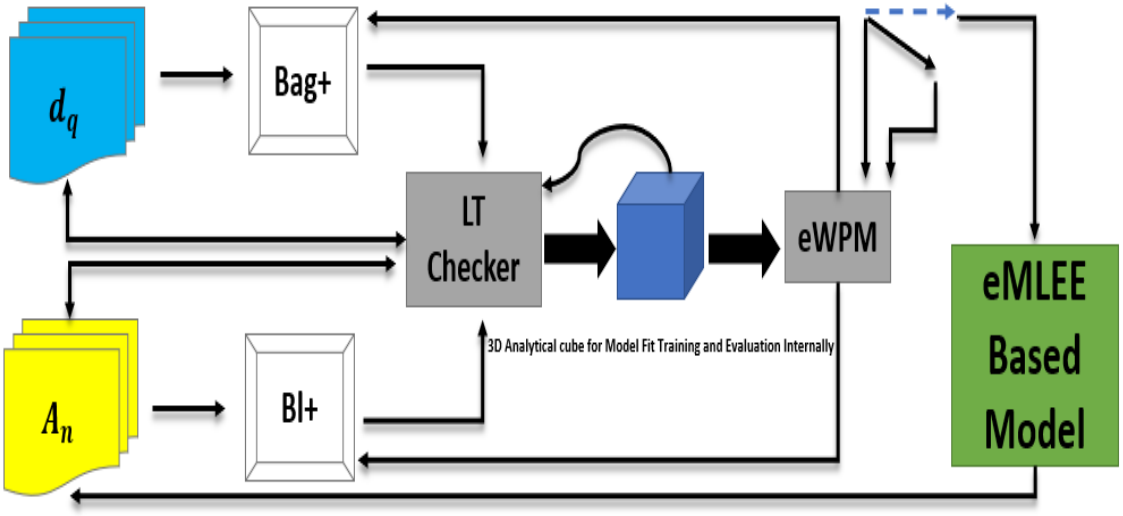


Figure 3.13– Engine Parallel Processing Internals

Fig 3.14 shows the internals of the *eCVS* construct and high level pseudo-code. Based on *LT* unit, it optimizes the value of k during split for train and test.

Fig 3.15 shows the internals of *eWPM* construct and high level pseudo-code. It assists in the improving the reliability of the accuracy measure. It other words, it tells the engine how accurate the accuracy is, so the real-world training can further be optimized and endorsed.

Fig 3.16 shows the internals of the eABT unit (Unit-A) and a high-level pseudo-code to represent the internal operation. This unit, as discussed in chapter, is solely

responsible for blending the methods for reaching the maximum possible model fit for improved generalization and reduced errors.

Fig 3.17 shows the internals of the eFES unit (Unit-B) and a high-level pseudo-code to represent the internal operation. This unit, like Unit-A is responsible for quantifying and grouping the right number of features for optimized training experience by end model.

Prepare d_q
Create CV Standard Split
Record the K in LT object
Compute the desired measure
While (New eABT.eWPM.F > eABT.eWPM.F) ***Do***
    Compute new K using Optimizer Function
    Update the LT object
    Re-split the d_q
    If (Threshold is not met for overhead) ***Then***
        Re-run the optimizer
        Reset the LT object
    Else
        Overwrite the new K
        Recompute the eABT.eWPM.F
        Update the LT object
    End If
End While

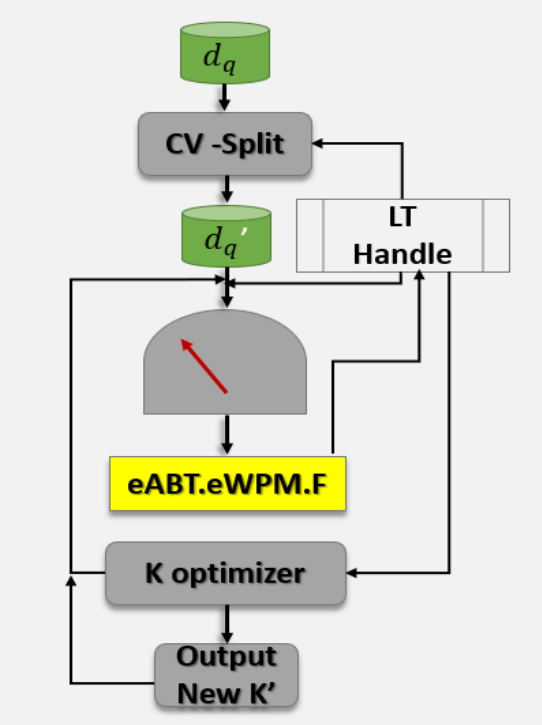


Figure 3. 14– Enhanced Cross Validation and Split (eCVS) Internals

Compute Confusion Matrix
Compute Standard Metrics
Store in LT object
While ACC' Improve ***Do***
Split n number of data sets as per LT rules
Compute ACC' = $\left[\sum_{i=1}^{L} \frac{(tp_i + tn_i)}{tp_i + fp_i + fn_i + tn_i}\right] / L$
Stop once LT reports threshold met
End While
Compute Internal metrics
Compute Avg_acc' for all new L'
Compute ACC'' = $\left[\sum_{i=1}^{L} \frac{(tp_i + tn_i)}{tp_i + fp_i + fn_i + tn_i}\right] / L'$
Compute the $\{Acc_{avg}^{n} = (Acc_{avg}\text{'})/n$

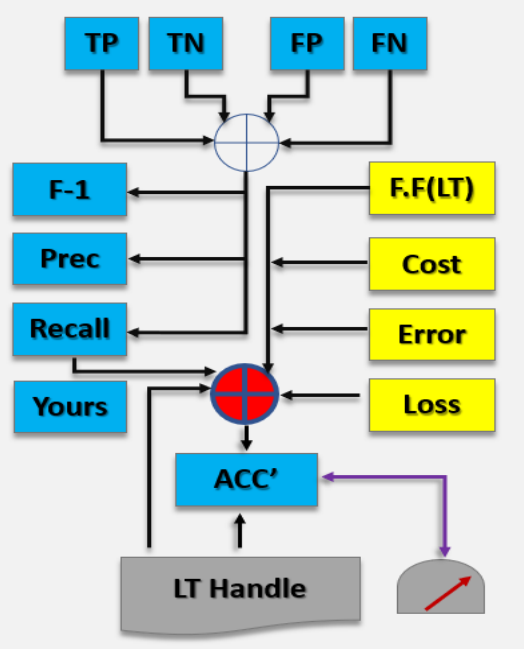


Figure 3. 15– Enhanced Weighted Performance Metric (eWPM) Internals

Prepare Datasets as d_q
Split the d_q Using D.Bagging+ approach
Store the List of all available SL Methods
Record the Highest and Lowest Errors Using LT objects
While (Error is in between above limits) ***Do***
Blend the Algorithm from the pool using LT handle
Compute MF using ALGO 3
Record MF value in LT object
Compute ALGO 4 using Error Rules
If (CF is above 50 %) ***Then***
Create a Pointer at Methods Blend in LT
Compute Gain Factor Using ALGO 5
Feed it back to ALGO 4
Else
Compute ALGO 3
Update the LT Pointer
End If
Re-Bag the Dataset to increase the error
Note the maximum error (mistakes) Using eWPM in LT object
Compute ALGO 6
Update it in LT object
End While

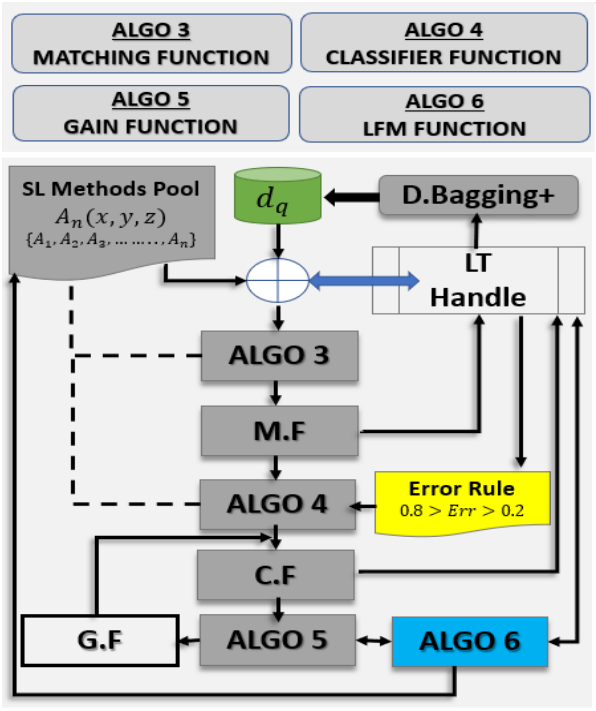


Figure 3.16 – Enhanced Blend and Tuning (eABT) Internals

Import all the available features
Randomly pick a subset and Run the ALGO 7
Compute FFF and Update the LT object
While (FFF is not flagged by LT Handle) **Do**
Compute $LT.RedF(F(n))$
Compute $LT.IrrF(F(n))$
Update Features Pool
End While
Compute FGF using ALGO 8
Compute eWPM, as features are added from the POOL
If (Metrics.Accuracy is improving) ***Then***
Keep Adding features
Record the measure
Else
Update the LT pointer
Add to the Waiting List
End If
Compute Feature Adder Function Using LT handle
Compute Feature Remover Function Using LT handle.

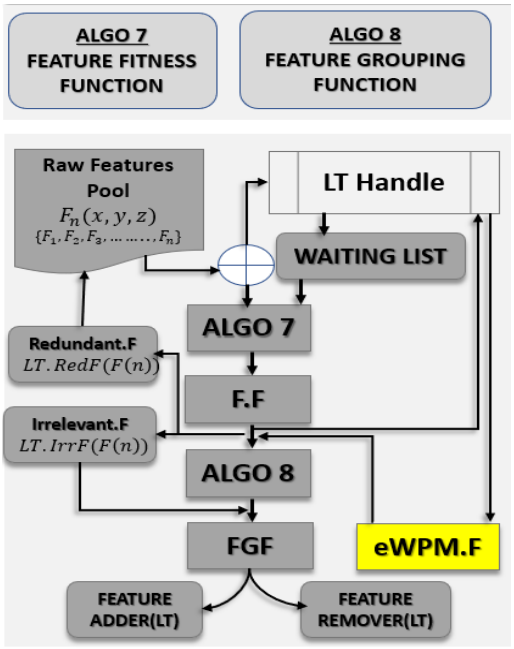


Figure 3.17– enhanced Feature Engineering and Selection (eFES) Internals

3.5.3 Aggregating C-UNIT, UNIT-A, UNIT-B

Fig 3.18 shows how the engine aggregates each unit to construct the final box as a eMLEE. The second figure shows the *3D* computation example. Instead of using only one metric such as overfit or underfit, eMLEE computes the FF based on three-dimensional analysis as shown. Eventually the goal of eMLEE is to compute the FF based on z-dimension that truly indicates the best reduced value of x(overfit) and y(underfit). Depending on how LT handles it, the $\{Z_i to\ Z_n\}$ is determined and appropriate matrix computation is done.

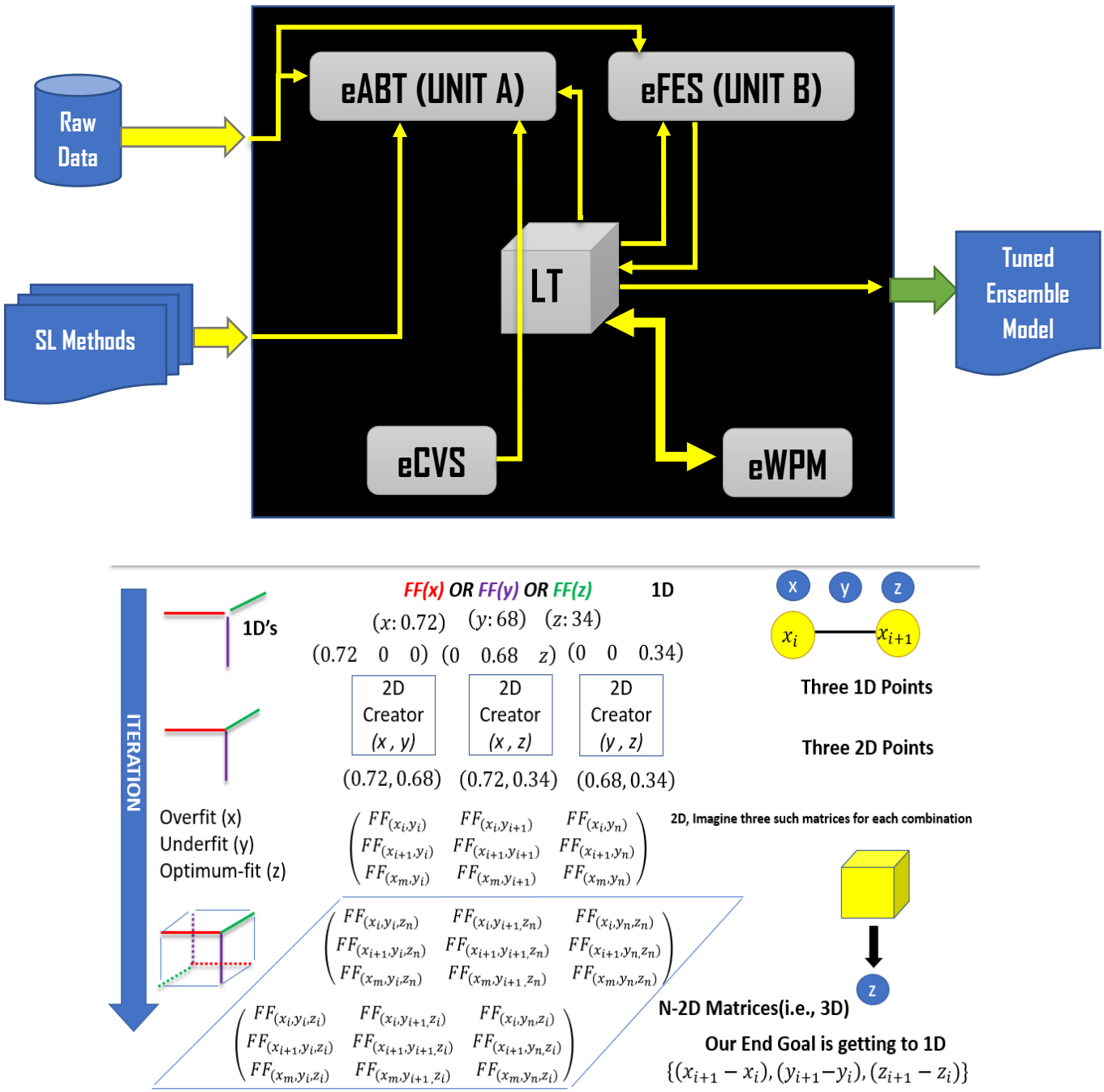


Figure 3. 18 - eMLEE Internal Structure and Illustrative example for 3D computation of fitness factor(FF)

CHAPTER 4 – INTERNAL eMLEE ALGORITHMS

4.1 LT Algorithms

4.1.1 Plain English Description

Algorithm 1 - The main *while* loop at **step 3** makes sure that rule 2 is obeyed. **Steps 4-6** compute error functions and Ratio as constructed in the math model. The first ***For*** loop at **step 7** ensures the optimum fitness is regulated in the z-dimension. **Steps 8-12** build the probability distribution hypothesis, so the cost function is decentralized for improved labeling of each element in each row as *LT* object receives it. *If* block at **step 13** checks and maintains the probability of the fitness to be greater than 50 % for more training to be continued and then we update the *LT* objects. **Step 22-28** sets the changes in each dimension for the algorithm element being incorporated and then updates the global object. We compute ERM function and use equation 5&6 to utilize adder and remover function. In **steps 29-35**, we update the *LT* objects for all the *ML* methods incorporated (added or removed) based on the desired fitness and error ranges as per rules defined in the module. **Steps 36-40** finally compute the blending and tuning functions as we constructed in the mathematical model and return the quantized data to the calling function of the algorithm object.

Goals: It governs *LT* structure in the memory to keep track of fitness of the model for algorithms (i.e., SL methods) blend.

Input: $A.P = \{A_1, A_2, A_3, \ldots\ldots, A_n\}$/*Pool of SL methods */
Output: $NODES_{i \in N}$, e$ABT^{\boxplus}$

Algorithm 2 - Step 1 initializes the optimum fitness factor. **Step 2** begins the ***While*** loop to check for Ratio that is governed by local and global errors correlation, so the function remains in-bounds of over-learning and under-learning logical *3D* space. **Steps 3-4** compute the global Error and hypothesis function for even probability distribution as discussed in the mathematical model. **Step 5** starts the ***For*** loop to evaluate each feature and quantifies *x,y, and z* as it spreads in space using *3D* logical elements.

Steps 6-20 compute the Ratio function so the local error can be regulated and then update the *LT* object in the library call. Then it resets each co-ordinate for next run in the loop. **Steps 21-25** build the references for computing Feature Adder and Feature Remover function for feature grouping function using *LT* parallel evaluation technique as explained earlier in the model build of Section 4. In **Steps 26-30**, the ***If*** block checks for each algorithm entry so the Ratio can be re-calculated and this way, the *No Free Lunch Theorem* problem is also addressed. Finally steps **31-35** compute the main *eFES* function after updating the central probability function, so the bias can be minimized for each feature before adding to the group. Then in last step, the function reference is returned to the calling pointer of the algorithm.

Goals: It governs *LT* structure in the memory to keep track of fitness of the model based on features

Input: $A.P = \{A_1, A_2, A_3, \ldots\ldots, A_n\}$ /* *The pool of SL methods**/
$F(x, y, z) = \{F_1 \in F_n\}$ /* *Features set* */
Output: $NODES_{i \in N}$, e$FES^{\boxplus}$

4.1.2 Pseudocode

Algorithm 1 –LT eABT GOVERNANCE

Initiate: Create data libraries object as *ObjDS*, *ObjLT*
Set: $x, y, z \leftarrow$ *ObjDS.RandomValues(0)*
While $(0.2 < err \in Err < 0.8)$ ***Do***
    Compute: error constant
    Set: $\mu \leftarrow \frac{1}{N} \sum_{i=1}^{N} (x, y, z)_i$
    Compute: $\mathbb{R}_{eABT}$
    For (each ObjLT.Evaluate(1) in *z*) ***Do***
        Set: z ← 0
        Compute: $T(m, a)$
        Set: $\gamma \leftarrow \frac{1}{2} \ln\left(\frac{1-\varepsilon_t}{\varepsilon}\right)$
        Update: Probability of Hypothesis:
        $H_t : I \rightarrow \{-1, +1\}$
        If $(P_{t+1}(k) < 0.5)$ ***Then***
            Set: $Err_t \leftarrow \sum_{k:H_t(i_k) \neq o_k} P_t(k)$
            Read: ObjDS.Evaluate(Err_t, γ)
            $P_{t+1}(k) \leftarrow \frac{P_t(k)}{Z_t} \times \begin{cases} e^{-\gamma t} & If\ H_t(i_k) = o_k \\ e^{\gamma t} & If\ H_t(i_k) \neq o_k \end{cases}$
            Compute: ObjLT.Write($P_{t+1}(k)$)
        Else
            Set: $Err_t \leftarrow ObjLT.Read(H_t, \gamma)$
            Update: ObjLT.Update(err, Err)
        End If
        Set: $A(x, y, z) \leftarrow (\Delta x + 1), (\Delta y + 1), (\Delta z + 1)$
        Compute: $ERM(3D)$
        Compute: Add/RemFunc
        Write: ObjLT.Write(*ERM(3D)*)
    End For
    Compute: ObjLT.FitnessScore(*A(i)*)
    Update: ObjLT.Update(err(z), Err(z))
    For (each node in $NODES_{i \in N}$) ***Do***
        If $(Score(A_i \in A_{(i+1)}) > node(i))$ ***Then***
            Update: ObjLT.Zscore(ObjDs, A_z)
            Read: ObjLT.Read(score(z))
        End If
        Set**:** Next node
    End for
    Compute: $\mathbb{B}_{A_n}$
    Compute: $\mathbb{T}_{A_n}$
    Update: ObjLT. e$ABT^{\boxplus}(\mathbb{B}_{A_n}, \mathbb{T}_{A_n})$
End While
Return: e$ABT^{\boxplus}$

Algorithm 2 – LT eFES Governance

Set: $\{Op.F \leftarrow 0, F(x.y,z) \leftarrow (0,0,0)\}$
While $((\mathbb{R}_{eABT}) < (r(e+E))$ ***Do***
    Compute: $Err_t \leftarrow \sum_{k:H_t(i_k)\neq o_k} P_t(k)$
    Compute: Hypothesis: $H_t : I \rightarrow \{-1,+1\}$
    For (# of F in Set) ***Do***
        Set: $\gamma \leftarrow \frac{1}{2}\ln\left(\frac{1-\varepsilon_t}{\varepsilon}\right)$
        Compute: x,y, and z for ObjLT.Random()
        Update: ObjLT.Update(*x,y,z,* γ)
        If $(\gamma < (\gamma - 1))$ ***Then***
            Set: $\gamma \leftarrow (\gamma + 1)$
            Read: ObjLT.Read*(x,y,z)*
            Compute: : $P_{t+1}(k) \leftarrow \frac{P_t(k)}{Z_t}$
        Else
            Set: $\gamma \leftarrow (\gamma - 1)$
            Update: ObjLT.Update(*x,y,z,* γ)
        End If
        Compute: $\mathbb{R}_{eFES} \leftarrow \frac{1}{N_e}\left(\sqrt{\frac{e}{e+E}}\right)^2$
        Update: ObjDS.Update($\mathbb{R}_{eFES}, ObjLT$)
        Set: $x \leftarrow (x+1), y \leftarrow (y+1), z \leftarrow (z-1)$
    End For
    Write: ObjDS.Write(*x,y,z, ObjLT*)
    Compute: $+\mathbb{F}$
    Compute: $-\mathbb{F}$
    Update: Scores for each algorithm, and creates Nodes $NODES_{i\in N}$
    For (each node in $NODES_{i\in N}$) ***Do***
        If $(SCORE(A_i \in A_{(i+1)}) > node(i))$ ***Then***
            Add: entry to LT
            Re-compute: $\mathbb{R}_{eABT}$
            Update: LT
        End If
        Finally Update: $P_{t+1}(k)$
        Compute: eFES$^{\boxplus}$ $(x,y,z) \leftarrow ObjLT.Optimize(\mathbb{R}_{eABT}, \mathbb{B}_{A_n}, \mathbb{T}_{A_n})$
    End For
End While
Return: $NODES_{i\in N}$, eFES$^{\boxplus}$

4.2 eABT Algorithms

4.2.1 Plain English Description

Algorithm 3 incorporates all the ML methods for testing the engine training phase. Using the diverse dataset, the model is taught to incorporate method by method and observe the over-fitness as regulated by LT object. Eventually, MF is computed with given mathematical construct such that the algorithm has higher MF as compared to any other during testing phase. This algorithm also efficiently computes two very important functions (i.e., Loss and Cost). These functions are later processed by LT objects throughout the internal layers of the eABT unit. **Steps 2-5** set the coordinates and split initial values. **Step 6** computes the Adder and Remover function as detailed in the math model earlier. **Step 7** begins the For Loop to compute the 2D array elements and distance function. **Steps 12-15** configure the LT objects in relevance of each coordinates. **Step 16** computes the derived function. This function is then monitored for quantification in **If** block at steps **17-21**. Finally, in **step 22**, the computation of MF begins. LT pointer is used to improve the learning process at this stage as we pointed out earlier.

Input: $A_n = \{A_1, A_2, A_3, \dots \dots \dots, A_n\}$, *Raw dataset* $DS(n)$
Output: $M.F\{0:1\}\ Matching$ function
Libraries: Create: *ObjeMLEE*(h) */*Create an object reference of eMLEE API */*
Initialize: *ObjeMLEE*.PublicFunctions(*h.eABT,h.eFES,h.eWPM,h.eCVS*) */* Handles for all four constructs*/*

Algorithm 4 aims to determine the classifier function of the eABT unit. It divides the data into two major samples and then apply the cross-validation techniques. As stated earlier in the introduction that eCVS is a sub-unit of the eMLEE engine that is a work in

progress to improve the validation process, where the algorithm such as presented here determines the value of k for the most optimized state of the learner during training. This greatly reduce the over-learning and biasing of the model. Finally, this algorithm 2 aims to maintain the **Rule. 2** guidelines and accuracy function during the formulation of the classifier function. The **While** loop at **step 1** governs the global condition for classifier function. **Step 2** sets the initial values for 3D coordinates. **Steps 3-5** computes the γ function so each coordinates become the logical objects in the space for LT recording during classifier learning. **Steps 6-9** validate it using API object. **Steps 10-18** compute Loss and Noise function using Blending function along with CV process. This way Rule 2 is validated, and LT regulates the changes in the metrics from each object in observed space. **Step 20** resets the coordinates and then LT computes the maximum accuracy achieved and based on it it re-computes the classifier function and return it to the calling function.

Goals: Based on MF, it computes Classifier function for the blend in progress.
Input: $M.F\{0{:}\ 1\}\ Matching$ function
Output: $(\psi(n))$, classifier function

Algorithm 5 does not return any output, but it is responsible for setting up the internal variables of the mathematical constructs and functions that the proposed model uses through *LT* objects. This algorithm 3 also govern the crucial mathematical constructs of the *eABT* unit. It works in conjunction with Algorithms 1 and 2 to provide the parallel construction of blending and tuning process as discussed earlier. As the main while loop runs, it is ensured that all *tp* values are in correlation with GG and gg. This further leads to optimization of the model as $gg \in GG$ develops to regulate the 3D models of each metric as discussed earlier. The condition defined by the construct as

$\{gg_{xn}, gg_{yn}, gg_{zn}, \ldots., gg_{n\in N}\} \in GG\{x, y, z\} \leftarrow LT.GAIN(gg, GG)$ mainly ensures the classifier fitness when blend is being engineered based the distance function for low variance and high distribution specially for lower dimensions. This ensure that optimum solution does not move only in local minima. There are two main loops at **Step 1** and **Step 24**. Loop 1 regulates m iterations for gain function computation for the given datasets. It must be noted that we seek the optimization logical points in *z*-dimension, as discussed in the model building process earlier. **Step 5** shows the multi-dim matrices computation for local gains as it correlates with global gain function, thus we check each index value in ***If*** block at **step 6**. Once that stabilizes in local for loop at **step 4**, we then create another for loop at **step 14**, which basically re-check (re-tunes the classifier) so we ensure the evaluation of each logical point in *3D* space as the classifier process continues. Finally, the ***repeat*** loop computes the distance function to mature the Gain function in *z* dim eventually and thus the logical table is updated in each run.

Goals: It regulates the *3D* model for Algorithm blend and tuning functions. It computes the crucial distance functions.
Input: Dataset, Algorithm pool, LT objects,
Output: Optimized z for the final blend.

Algorithm 6: It monitors the metrics, such as *U.F. O.F. B, error, Accuracy* and create a logical data structure (*DaSr*). This *DaSr* is used to feed back the poor prediction of the model during testing so it can learn from its mistakes during next training phase. This algorithm further regulates the LFM based training as modeled in Definition 6 earlier.

Steps 25 - 31 plays very important role in the algorithm definition to teach the model when to stop the learning so the bias and outlier's errors can be minimized.

Goals: It aims to segment the incorrect predictions to be used for classifier learning.
Input: Dataset, Algorithm pool, LT objects
Output: Optimized z for the final blend.

4.2.2 Pseudocode

Algorithm 3 – eABT Matching Function Quantification

While **(**FF **(**A_n**)** > A_{n-1}**)** ***Do***
    Set: $x \leftarrow 0, y \leftarrow 0, z \leftarrow 0$
    Set: $LTobject[x, y, z]$
    SetErrorBounds**:** $\lim_{LE} \leftarrow UB(0.8), LB(0.2)$
    Create: *Test and Train Split using h.eCVS*
    Compute: AddFunc, RemFunc
    For each ($A_i \in A_n$) ***Do***
        Update: $\mathfrak{C}(x, y)$
        Compute: $\nabla d(A_1, A_2)$
        Execute: h.Addrow(A_i)
    End For
    Update: $LTobject([x, y, z], h)$
    Read: *LT(z)* into z'
    Set**:** $\{(x_i \leftarrow x_{i-1}), (y_i \leftarrow y_{i-1}), z \leftarrow z'\}$
    Compute: $\mathbb{Q}$
    If **(**$\mathbb{Q} < 0.5$**)** ***Then***
        Compute: $L(x, y, z)$
        Compute**:** $C(x, y, z)$
        Update: h.Update(*L*, *C*)
    End If
    Compute**:** $M.F\{0:1\}$
    Reset: $x \leftarrow 0, y \leftarrow 0, z \leftarrow 0$
    Update: LT.Save(*x*,*y*,*z*, *MF*)
    Read: $FF \leftarrow LT*$
End While
Return *MF*

Algorithm 4 – eABT Classifier Function Computation

While **(**$\psi(n)) < 0.5$ && $Rule\ 3\ return\ '1'$ **)** ***Do***
    Set: $x \leftarrow 0, y \leftarrow 0, z \leftarrow 0$
    Compute: γ
    Set*:* $X \leftarrow (*, \gamma, \gamma), Y \leftarrow (\gamma, *, \gamma), Z \leftarrow (*, *, \gamma)$
    Compute: $\|M\|(x, y, z) \leftarrow \sqrt{\sum_{x=1}^{X} \sum_{y=1}^{Y} \sum_{z=1}^{Z} M_{x,y,z}^2}$
    If *(*$\|M\| \geq 0.5\ for\ each\ (x, y, z)$) ***Then***
        Compute: $\psi(n) \leftarrow \underset{(x,y,z \in Z)}{\text{Extract}}(\ \|M\|(x, y, z)\ ) + \log \sum_{k=1}^{Z} \gamma^{z+1}$
        Update: ObjeMLEE.Update ($\psi(n)$)
    End if
    Reset: *LT* Pointer
    Re-compute: $L(DS(S(x, y, z), N(x, y, z)$
    Run: Comparison of old and new *L*,*N* Function
    If (Rule 2 is True) ***Then***
        Update: $\mathbb{B}(x, y, z) \leftarrow [L(x, y, z]^i$
        Increment: *i*
        Update *LT.Addrow(i)*
        Create*:* h.eCVS($S_1, S_2, \ldots\ldots, S_n$)
        Check: LT.Rule(#2)
    End if
    Reset: $x \leftarrow 0, y \leftarrow 0, z\ 0$
    Update: Classifier Function for Binary *'1'*
    Update: LT.Accuracy (*tp*,*tn*,*fp*,*fn*)
    Recompute: $\psi(n)$
    LT.Update (*h**)
End While
Return $\psi(n)$

Algorithm 5 – Gain Function Optimization

While (*i = 1 to m*) ***Do***

$x \leftarrow 0, y \leftarrow 0, z \leftarrow 0$

$x_{ij}, y_{ij}, z_{ij} \leftarrow LT.MATRIXCLASSIFIER$

For (*j=1 to n*) ***Do***

$$\begin{Bmatrix} gg_{x1} & gg_{x2} & \cdots & gg_{xn} \\ gg_{y1} & gg_{y2} & \cdots & gg_{yn} \\ gg_{z1} & gg_{z2} & \cdots & gg_{zn} \end{Bmatrix} \leftarrow GG \times \begin{Bmatrix} z(i,j) & \cdots & z(i,n) \\ \vdots & \ddots & \vdots \\ z(n,j) & \cdots & z(m,n) \end{Bmatrix} as\ MM$$

If $(MM(j) \geq M(x) + M(y))$ ***Then***

Set: $M(z) \leftarrow MM(j)$

Else

Set: $MM(j) \leftarrow MM(z)$

End If

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

End For

$\alpha\beta\gamma \leftarrow \{0,0\},\{0,0\},\{0,0\}$

For k = *1 to LengthOf(MM)* ***Do***

Compute: $GG \leftarrow \sum_1^k \alpha + \sum_1^k \beta + \sum_1^k \gamma$

Update: $LT.GAIN \leftarrow gg \in GG$

Update: $\alpha\beta\gamma \leftarrow \{k, GG\{x\}\},\{k, GG\{y\}\},\{k, GG\{z\}\}$

Set: $\widehat{m_k} \leftarrow c(F_k, \vartheta)$

End for

$S_{input} \leftarrow \{1,2,3,4, \ldots \ldots, T_n\}$

$S_{output} \leftarrow \{1,2,3,4, \ldots \ldots, T_n\}$

AF $\leftarrow A^{z \in Z} \times \left[\prod_{i,j,k}^{N \times M} \Delta_{(x,y,z)}^{(a,b)}\right] Prob\{Y_i = y_i\}$

End While

Repeat

$\{gg_{xn}, gg_{yn}, gg_{zn}, \ldots, gg_{n \in N}\} \in GG\{x,y,z\} \leftarrow LT.GAIN(gg, GG)$

If $x(gg) + x(GG) \neq (x-1)gg$ ***Then***

Update: $E.D \leftarrow \sqrt{(x_1 - x_2)^2 + (y_1 - y_2)^2}$

Update: $H.D \leftarrow \sum_{j=1}^{k} |x_j - y_j|$

Update: $M.D \leftarrow \left(\sum_{j=1}^{k} |x_j - y_j|^p\right)^{\frac{1}{p}}$

End If

Until *GG and gg Correlates with TP*

Algorithm 6 – Learn From Mistakes Function Computation

1: ***Repeat***
2: Extract: *LT*.Gain(*GE*, *LE*)
3: Set: $x \leftarrow 0, y \leftarrow 0, z \leftarrow 0$
4: Compute : $LE(OOB) \leftarrow \prod_{=1}(\widehat{x_{t0}^{N}} - \widehat{y_{(-i)}})$ Compute:
5: $LE(IB) \leftarrow \prod_{=1}(\widehat{x_{t0}} + \widehat{y_{t0}^{N}})$ Process: *LT*.Process
6: $(LE(OOB), LE(IB))$
7: ***For each*** $(LE(x, y, z))$ ***Do***
8: Set: $\widehat{x_t} := w + x_t y - y_t x$
9: Set: $y_i \leftarrow x + y_t$
10: Set: $\widehat{y_t} := max\{< w, err(z) > err(x, y)\}$
11: Compute *AF* as per (35)
12: ***If*** (*AF*>0) ***Then***
13: Update: *LT*.Fitness (AF)
14: ***End If***
15: ***End For***
16: ***For*** $(tp, fp, tn, fn \in Valid\ Flags)$ ***Do***
17: Compute: $\langle \mathbb{H}(x, y), \mathbb{H}(z) \rangle$ per (38)
18: Update: *LT*.LFM$(tp, fp, tn, fn) \leftarrow \mathbb{H}$
19: Run: Binary Classification Test on each data segment
20: Update: the LT arrays
21: Compute *LT*.LowerLayer$(\zeta(tp, fp, tn, fn))$
22: ***End For***
23: Reset: $x \leftarrow 0, y \leftarrow 0, z \leftarrow 0$
24: Check the Conditions using LT.*Check*
25: ***If*** *(If* $GG \geq \|\mathbb{W}^n - 1\|$ *) AND (gg{z} > 0.5))* ***Then***
26: Add: *DaSr* ← *LT*.Addrow(Lines 21 to 26)
27: Update: $Risk_{emp}[f'] \leftarrow GE(z)$
28: Compute: $\phi_{x.y.z} \leftarrow Pr(gg(z)) = 1$
29: Copy: $x_i \cong y_i \rightarrow y_i \cong x_i$
30: Copy: $\alpha\beta\gamma(\zeta) \rightarrow [\sigma(x_i, (y_i), (z_i))]$
31: ***End If***
32: ***Until*** (*LT*.Check(Evaluate(Row{i},*i++)*

4.3 eFES Algorithms

4.3.1 Plain English Description

Algorithm 7 aims to compute the low-level function as *F.Prep (x,y,z),* based on final Equations as developed in the model earlier. It uses the conditions of Irrelevant feature and Redundant feature functions and runs the logic if the values are below 50% as a check criterion. This algorithm splits the training data based on popular approach as cross validation. However, it must be noted in **step 6**, that we use our model API for improving the value of *k* in the process, that we call enhanced cross validation. LT object regulates it and optimizes the value of *k* based on the classifier performance in the real time. It then follows the error rule (80%, 20%) and keeps track of each corresponding feature, as they are added or removed. Finally, it gets to the start using the gain function in 3D space for each fitting factor since our model is based on 3D scoring of each feature in the space where point is moved in *x*, *y*, and *z* values in space (logical tracking during classifier learning).

Input: Sample Dataset (ΔS_n)
Output*:* F.Prep (x,y,z)

Algorithm 8 aims to use the output of algorithm 1 in conjunction with computing many other crucial functions to compute a final function of feature grouping function (FGF). It uses the weighted function to analyze each participating feature including the ones that were rejected. It also utilizes the LT object and its internal functions using the API. This algorithm slices the data into various non-overlapping segments. It uses one

segment at a time, then randomly mixed them for more slices to improve the classifier generalization ability during the training phase. It uses eFES$^{\boxplus}$ as a LT object from the library of eMLEE and records the coordinates for each feature. This way, entry is made in LT class, corresponding to the gain function as shown in **steps 6-19**. From **steps 29-35**, it also uses probability distribution function, as explained earlier. It computes two crucial functions of $\nabla(\varphi, \rho, \omega)$ and $\mathbb{G}(x, y, z)$. For this global gain (GG) function, each distribution of local gain $g\ (x, y, z)$ must be considered as features come in for each test. All the low probability-based readings are discarded for active computation but kept in waiting list in the LT object for the second run. This way, algorithm does justice to each feature and give it a second chance before finally discarding it. The rest of the features that qualify in first or second run, are then added to the FGF.

Input: Sample Dataset (ΔS_n), F.Prep (x,y,z)
Output: h. FGF

4.3.2 Pseudocode

Algorithm 7 - F.Prep (*x,y,z*) Feature Preparation Function

While (GG (x,y,z) < 0.5) ***Do***
    Compute: $P(CRV) \leftarrow \sum_{i=1}^{N} P_H(h_i)p\ \ CRV \mid DRV$
    Set: x ← 0, y ← 0, z ← 0
    Compute: err and Err (*x,y,z*)
    For ($F\ = \{F_1, F_2, F_3, \ldots\ldots,\ F_n\}$ ***Do***
        Apply: Cross Validation on $DS(sig, noi)$
        Update: The Split Function using h.eCVS (k, F)
        Set: $Q^N\ \rightarrow\ DRV$/* Based on Mapping function */
        Compute: $h.MI(Irr.F(x,y,z)|f_i, f_{i+1}), L.Func(z), L.Cost\ (z)$
        Update: h.MI (F_i)
        If (err is in bounds as per rule) ***Then***
            Mark: the feature F_i) and Flag.
            Update: each f $\in$ F $^{(n)}$, for which $f_n \geq F\{0.85,0:1\}$ is valid
            Select: F (n) based on random function
            and distribution in space: D[$f\ (T)|F^{(n)}\ \in\ \partial F\ (F^{(n)})$]
            While (Irr.F $\geq$ 0.5 $AND\ Red.F\ \geq 0.5$) ***Do***
                Compute:LTObject.Weighted (eFES$^{\boxplus}$, h. MI (F_i))
                Extract: MI − i ← I MI Index
                Set: $Irr.F\ \leftarrow\ \sum_{i,j}^{K} \begin{Bmatrix} f_{ii} & f_{ij} \\ f_{ji} & f_{ji} \end{Bmatrix}$
                Compute: MI for Entropy, CF as correlating factor
                Re-compute: MI and Red.F (MI)
            End While
        End if
        Set: $F^{(n)} = Constant\ value$ for Initial test
        Compute: F.Prep (x,y,z) ← h.blend (MI,z)
    End for
    Slice: Data Samples $\{\Delta S_n\ \in\ S\}$
    Compute: and Create Matrices
    Set: $\mathbb{G}\ (x,y,z)\ \leftarrow\ \frac{1}{N} \sum_{i=1}^{n} \{(g_i) \times MH$
    $Entropy\ (F_{n\in\ x,y,z})\ \leftarrow\ \sum_{t\in T\ (x,y,z)} \frac{|\ F_{(t:x,y,z)}|}{|F_t|} Entropy\ (F_n)$
    Compute: $Gain\ (I, F_{(t:x,y,z)})$
    Compute: gR (z)
End While
Reset: x,y,z
Update: h.Update (gR (z), h.Prep (x,y,z), CF)
Compute: F.Prep (x,y,z) ← h.Model (h*)
Return: F.Prep (x,y,z)

Algorithm 8 – Feature Grouping Function (FGF)

While $((W(\emptyset)\{\nabla(\varphi,\rho,\omega),1\}))\neq 0,\{\in 0,1\})$ ***Do***
    Slice: Data Samples $\{\Delta S_n \in S\}$
    Compute: $W(\emptyset)$
    $Y \leftarrow (x,0,z), X \leftarrow (0,y,z), Y \leftarrow (x,0,z)$
    Execute: h.Train ($\{\Delta S_n\}$)/* Sample training begins on data set */
    If (h.Train $\leq$ ABS $(\nabla(\varphi,\rho,\omega))$) ***Then***
        Compute: h.Biasness $(W(\emptyset), Prep\ (X,Y,Z))$
        Compute: $\widehat{\mathbb{Q}_\gamma}\ (\Delta)$
        Update: h.Record (LTObject (eFES$^{\boxplus}$, $\widehat{\mathbb{Q}_\gamma}\ (\Delta)$)
    Else
        Update: h.Record (h.Train)
        Set: $Y \leftarrow (x,0,z), X \leftarrow (0,y,z), Y \leftarrow (x,0,z)$
    End If
    Compute: h.localgain and h.globalgain
    For (g $\in$ $(g+1, \Delta Gain_I(w))$ ***Do***
        Compute: $Gain_I(w)$
        Set: $Norm.\ Gain_I(w)$ to local minima
    End For
    Compute: (F. Sco (x, y, z)) as h.execute (FF as fitness factor),
    + $(Gain_I(w), err, Err, F.\ Eng\ (x,y,z))$
    Compute: (F. Opt (x, y, z)) as h.concatenate ($F.\ Eng\ (x,y,z)$, Y,X,Z, $F\ (n)$)
    If ($gR\ (z) < 0.5$) ***Then***
        Compute: err (z) and Err (x,y,z)
        Update: H.RecordErrors (err,Err, gR (z))
    End If
    Execute: h.Update (gR (z), $\mathbb{G}\ (x,y,z)$)
    Update: LT function, LT.Gain (h*)
    For (all tests in $P\ (rV_i \mid pE) > P\ (rV_j \mid pE)$) ***Do***
        Re-compute: the LG and GG
        Update: the h.LT (P)
        If (P (pE | rV_i)P (rV_i) > 0.5) ***Then***
            Compute: $\nabla\ (\varphi,\rho,\omega)$
            Compute: $\mathbb{G}\ (x,y,z)$ for all distributions of g (x, y, z)
        End If
        Compute: (F. Sco (x, y, z))
        Compute: (F. Opt (x, y, z))
        Compute: h.FGF (F.Sco,F.opt)
    End For
End While
Return (h. FGF))

CHAPTER 5 –EVALUATION, RESULTS, AND ANALYSIS

5.1 Introduction

5.1.1 End Goal

We aimed to test our proposed engine (i.e., *eMLEE*) both at internal and external layers to validate its stability, integrity, and the real-world performances specially as compared to the existing techniques. To accomplish this goal, we used labelled datasets suitable for classification problems (i.e., 1 or 0). This thesis reports 20 datasets that we used to report some of the results in this book. We also list performance and comparison standard metrics that we selected to experiment and test our engine with. With such expansion of internal and external testing, we validated the outcome of the proposed engine to meet the industry expectations.

5.1.2 Evaluation Approach

We inimitably adopted the following approaches to make our experiments more reliable, easy to interpret, reproduce, and analyze for the model's validation, integrity, and evaluation.

1. We conducted several experiments to cover wide range of datasets that helped achieved in-depth training of the model to study various ranges of metrics. We then re-evaluated our math constructs and algorithms to improve engine internals. This way, our math

constructs governed by our algorithms, ensured the integrity of the model through the lens of real-world data and testing.

2. We used Python and R data analysis packages to test our algorithms. We used Prism, SigmaPlot, and Excel to produce our results. We uniquely adopted the approach of *3D* to have more observational value to our analysis for the proposed engine.

5.1.3 Experimental Setup

The various datasets were used to improve the generalization of the model. The details of datasets are listed in Appendix. Datasets were divided in three sections, as a standard practice, *i) Train, ii) Test, and iii) Validation*. However, we also uniquely split the data (as defined in the algorithm) governed by the real-time metrics using *LT* object. In this process, the random slices of data were created and then they were flipped to elevate the predictive errors temporarily. This way, *LT* objects learn on maximum possible errors and then tune itself (i.e., algorithm) to improve the slice in the next run and so on. This is also supported in *LT* mathematical model (i.e., Definitions). This also chains the ideas of enhanced validation and parallelism as we stated in the Introduction section.

Validation datasets split tested how well the model was learning (i.e., learned skill) and testing datasets substantiated the bias of the model, as it learned. In main model of *eMLEE* several *ML* algorithms such as Support Vector Machines, Decision trees, Logistic Regression, Bayes networks, etc., were used to test the model via the blending mechanism (i.e., *eMLEE* internals). However, the *eMLEE* underlying proposed algorithms allow researchers to incorporate any supervised learning algorithm of their choice, to overcome

the challenge of "No Free Lunch theory" as we discussed in the introduction section. We have used existing libraries of Python and R scientific packages on the exact datasets that we setup for our experiments, so we could draw comparison charts and record tabular data.

5.1.4 eMLEE as an Engine Testing Approach

After the development of the *eMLEE*, it became imperative to test it for real-world integrity and stability. It is also known as outer layer of the engine testing. However, while improving the fundamental of ensemble learning via proposed *eMLEE* engine, we did vital internal testing of various internal functions and measures to monitor, regulate, and validate the inner layers of the engine during development. They are known as internal or quality assurance testing during development /testing phases. Such testing was also required to improve the mathematical constructs and internal governing algorithms to aim for optimized model.

5.2 LT Unit Internal Testing

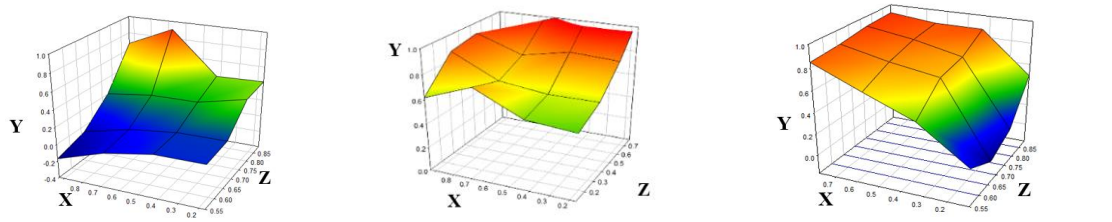
LT centralized unit as explained thoroughly in section 3 is responsible for the following main tasks at the lowest layers.

i) Computing x, y, and z during classifier blending learning and updating the table accordingly for overriding values.
ii) Computing eWPM metric to keep track of engine development performance.
iii) Aggregating the optimum set of features while quantizing the predictive value for each feature.

iv) Computing several other internal functions as explained in mathematical constructs and algorithm definitions.

With these points in mind, we hereby present some of the worth-noting results, observation, and discussion.

Fig 5.1 (a). shows the *LT* optimum fitness ability in each dimension. We noticed error at the negative value. **Fig 5.1 (b)**. shows the ideal behavior of the *LT* optimum fitness function. As we see, the blue section is virtually absent. It further elaborates that z-dimension has the maximum convergence of the function, as ideally desired. **Fig 5.1(c)**. is the real (experimental) behavior of **Fig 5.1(b).**



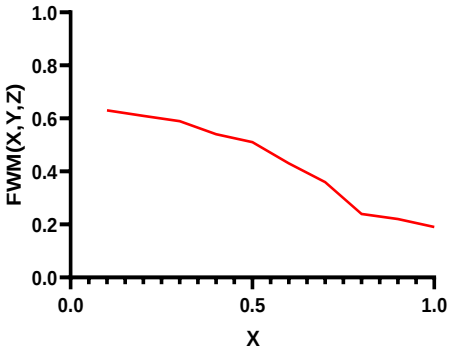
a) - It shows the *LT* optimum fitness ability in each dimension. We noticed error at the negative value.

b) - This shows the ideal behavior of the *LT* optimum fitness function. As we see, the blue section is virtually absent. And it further elaborates that z-dimension has the maximum convergence of the function, as ideally desired

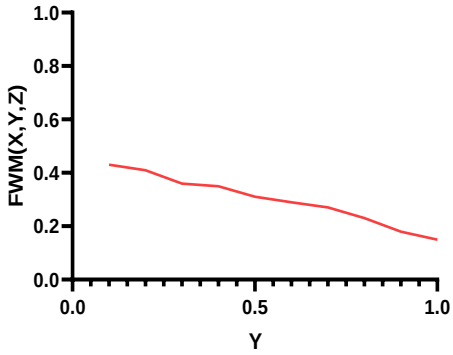
c) This is the real (experimental) behavior of Figure 32.

Figure 5.1 – Fitness Observation in 3D for Experimental Verification

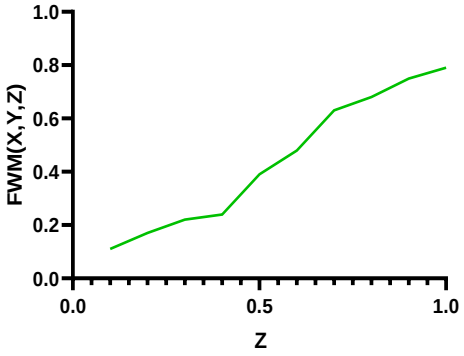
Fig 5.2. Shows FWM function response for various testing methods as discussed earlier.



(a) Here Y = 0 and Z = 0 were set and then the FWM was observed on X as overfit flag for the testing.

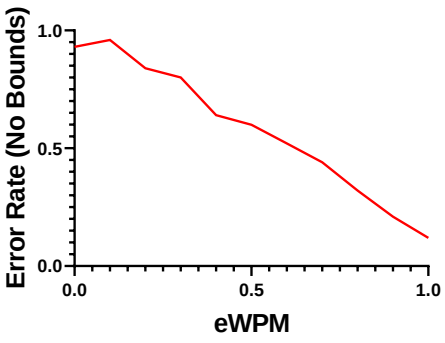


(b) Here X = 0, and Z = 0 were set.

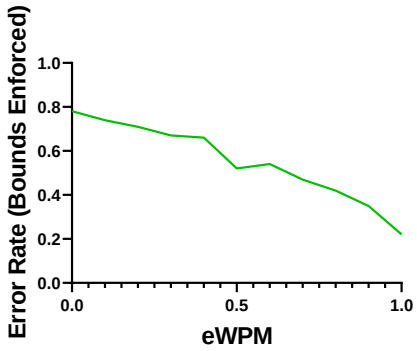


(c) This shows the stable response of FWM function for the dimension Z for valid values of z as it can be observed that FWM responded as expected when the engine was moving into optimum space as discussed in section 3.

Figure 5.2– FWM Internal Validation



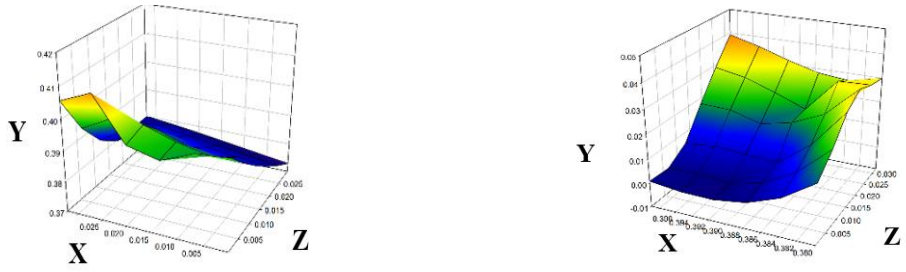
(a) The errors bounds were optioned out; thus, the errors were at high and lower points showing it is prone to overfit.



(b) Errors bounds were applied, and errors were not allowed to go over 80 % or drop below 20 % to tune it for optimum fitness.

Figure 5. 3 - Errors bounds observance for eWPM metric

Fig 5.4 (a) - This illustrates the ideal outcome of eFES LT fitness function in 3D space. Notice the z-axis has the least blue color. **Fig 5.4 (b)** illustrates the real(experimental) analysis of the test, we performed on validation *eFES* module. It should be noted that LT will create several (perhaps 100's) of such views. We have included sample results to support our arguments to develop reader's understanding.



(a) (b)

Figure 5. 4– eFES LT objects in 3D

Table 5.1, 5.2, and 5.3 show the typical results of the *3-D* measures. The snapshot of the results shows the difference of model behavior for theoretical (what we thought it will be) and experimental (what it turned out to be). These results are reported to support the model's stability in real world data with testing data.

Table 5.1 - x observations

Dim	Theoretical	Experimental
CF	0.23	0.44
Err	0.90	0.81
Acc	0.83	+0.76

Table 5.2 - y observations

Dim	Theoretical	Experimental
CF	0.17	0.53
Err	0.89	0.82
Acc	0.77	0.61

Table 5.3 - z observations

Dim	Theoretical	Experimental
CF	0.98	0.87
Err	0.25	0.31
Acc	0.95	0.87

In **Fig 5.5**, experiments (a) to (c) shows the poor optimization for z with CF without using LT objects. However, we observe in (d) to (f) that model is learning to optimize itself for optimum CF for z-dimension using LT objects. The spike noticed in (f) is suspected to be

error and will need future investigation. (a) through (c) were conducted using standard procedure where *'LT'* objects were not used.

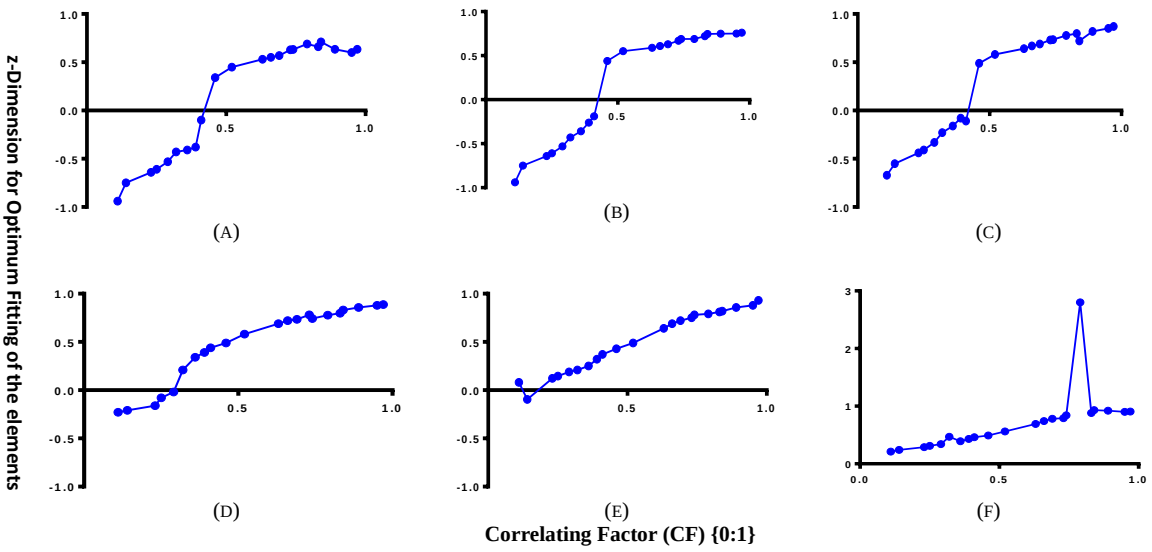


Figure 5. 5– Observation of poor optimization for z without LT object (Figure A to C). Improved optimization when LT object were incorporated (Figure D to F).

Fig 5.6(a) exhibits the model behavior for ratio ℝ in terms of each dimension of *x,y and z*. It is observed that regression is relatively higher for each dimension and is considered pre-mature learning of the model. **Fig 5.6(b)** shows improvement and considered mature classifier learning of the *LT* process. As we noticed that z-dimension (as hoped in the design of the model) is depreciating with respect of the error ratio ℝ. **Fig 5.6(c)** shows Adder real and ideal function is shown here. We observe that when experiment size is at lower end, it shows higher % and as the experiment size increases, the function outcomes drop, and this behavior is in line with model internals as expected. The triangular spike is a training error. **Fig 5.6(d)** shows Remover real and ideal function is shown here.

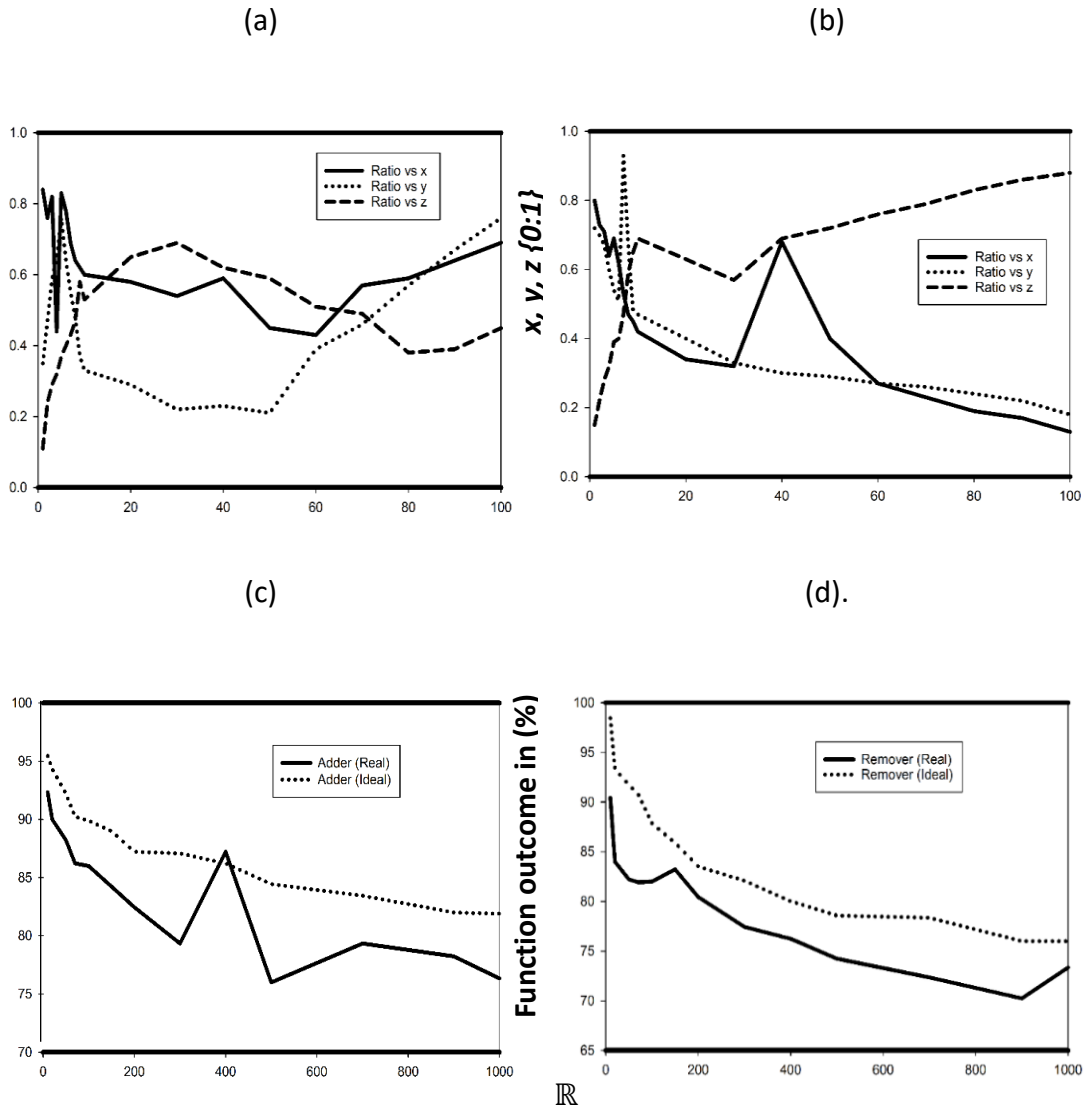


Figure 5. 6 - Ratio Function for Adder and Remover

5.3 eABT Unit Internal Testing

Fig. 5.7 shows the four simulations of the values as the function processes. It shows the expected correlation of incorrect predictions and correct predictions during the sampling period. *i)* $\sigma(w_{x,y,z})^2 + GG(z) < -0.5$ correspond to the observation when wrong prediction caused low variance and total dominance of the classifier function in z-

dimension, *ii)* $\sigma(w_{x,y,z})^2 - GG(z) < -0.5$ corresponds to false negative prediction based on feedback, *iii)* $\sigma(w_{x,y,z})^2 + GG(z) > -0.5$ corresponds to the optimum learning based on the feedback and low variance, and *iv)* $\sigma(w_{x,y,z})^2 - GG(z) > 0$ corresponds to very low variance and high GG correlation even for false positive markings.

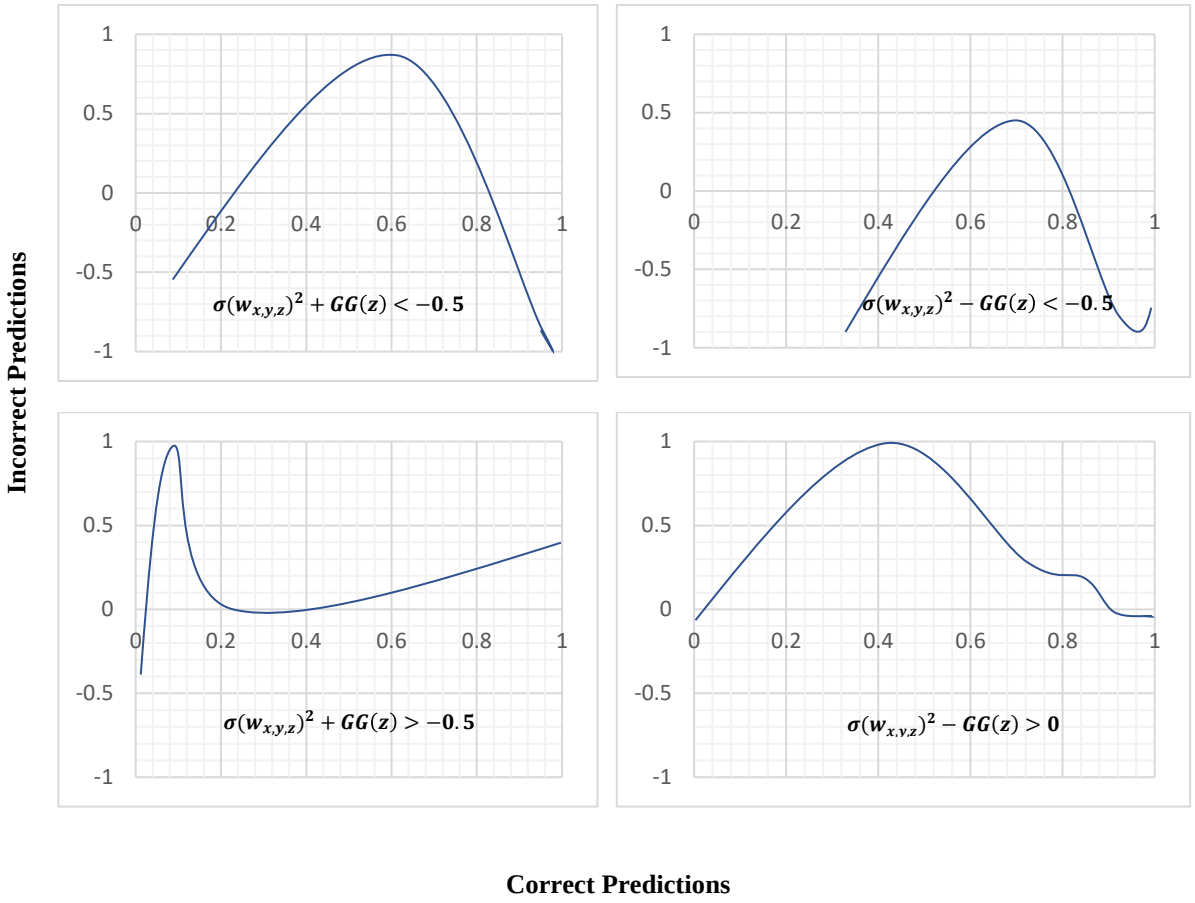


Figure 5. 7 - Experimental demonstration of the feedback mechanism

Fig 5.8 and 5.9 show the testing MF and CF Function as discussed in Chapter 3. It shows that as the number of iterations increased internally, the MF and CF function shows acceptable stability.

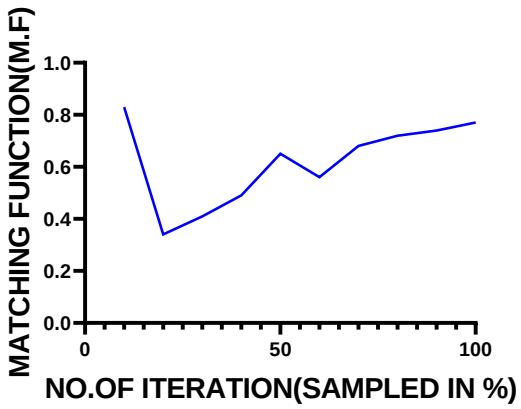


Figure 5. 8 - Matching Function Validation

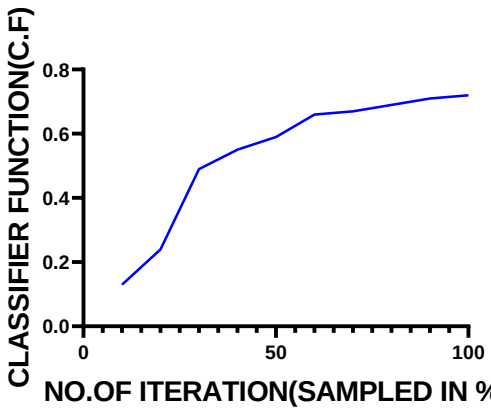
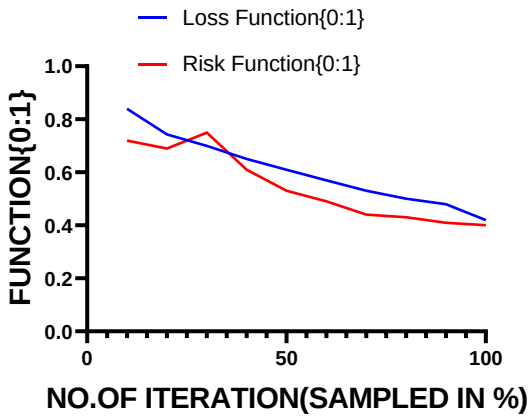


Figure 5. 9 - Classifier Function Validation



Loss function depends on the data we select. But we also want a different measurement which is an averaged over many possible data sets, and that is Risk: Average measure of loss.

Figure 5. 10 - Experimental demonstration of the feedback mechanism

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. Fig. 5.11 shows the 3D modeling of *eABT FF* with spreading of coordinates. Simulations were done in SigmaPlot software and Microsoft Excel.

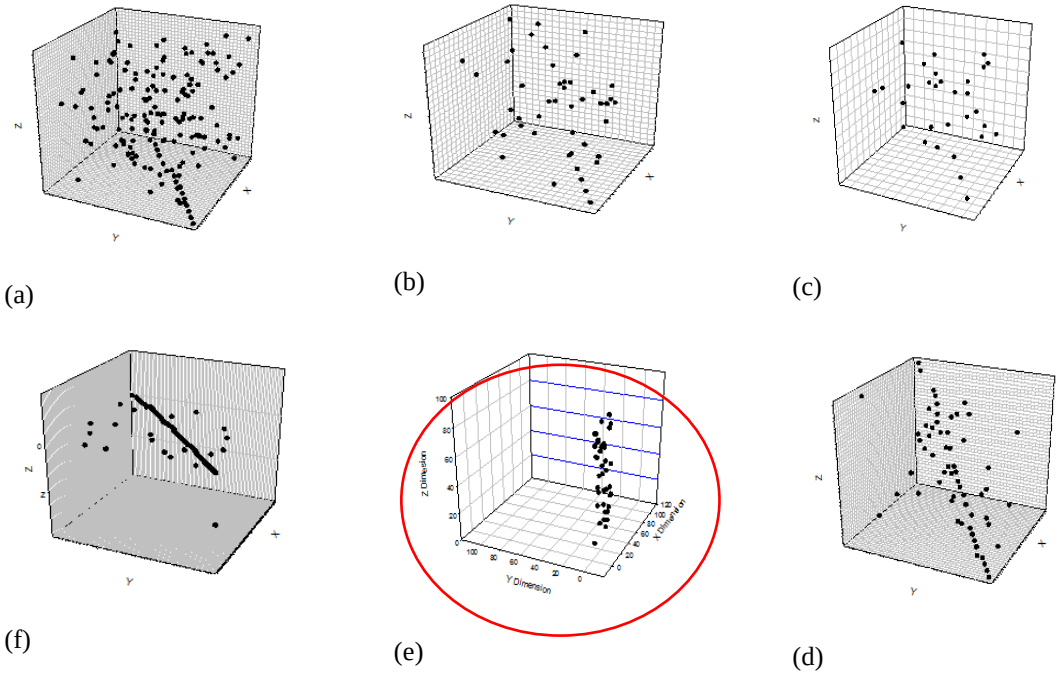


Figure 5. 11 - 3D modeling of eABT FF with spreading of coordinates.

(a) shows the high dispersion of all fitness functions. (b) shows the identification process has begun; (c) shows that the variables that are contributing to overfitting are filtering out. (d) shows the data points are streaming as expected; (e) shows the improved version of (d). (f) finally it is observed that x,y dimensions have reduced, and data has been filtered to be in the optimum fitness range based on tuned and enhanced algorithm/model, eMLEE.

Figure 5.12 - Legend:

x-axis : Shows the random data set mix for 40 experiments (training and testing)

y-axis: % predictive fitness. We used what are known as Stock/Candle stick type of charting for demonstrating the outcome of the model. Visuals from (a) to (d) shows different predictive patterns for the blended model, we proposed. It must be observed that for a random pick, the model fitness and generalization was observed to be acceptable for

the metrics values, as expected and engineered. The shaded black bars indicate the uptrend towards higher fitness function as expalined in section 3 and white (transparent) bars/candle sticks indicate downward trend. Longer the stick shade is, shorter it took to train and maintain the fitness function above threshold, as explained in chapter 3.

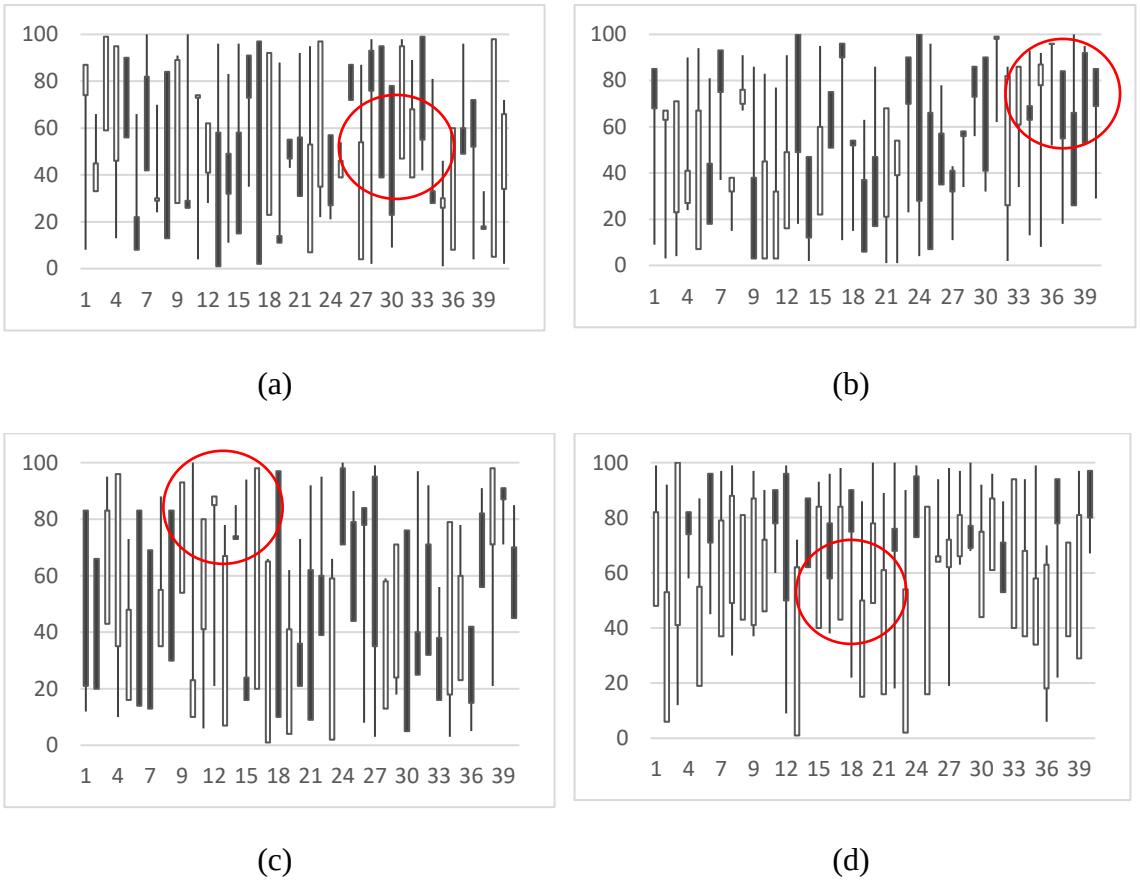
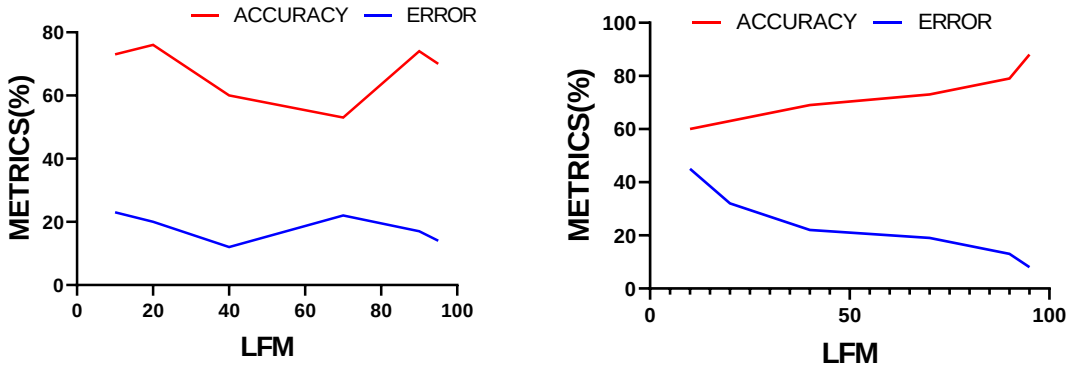


Figure 5. 12- Stock/Candle stick type of charting for demonstrating the outcome of the model

Fig 5.13 (a to b) show the internal testing of the working of LFM Function for Accuracy and Error observance.



(a) Shows poor tuning of the function as it exhibits poor correlation. It also indicates theta LFM function could not provide accurate feedbacks.

(b) It shows improved correlation and expected correlation.

Figure 5. 13– eABT' LFM Function Internal Testing.

5.4 eFES Unit Internal Testing

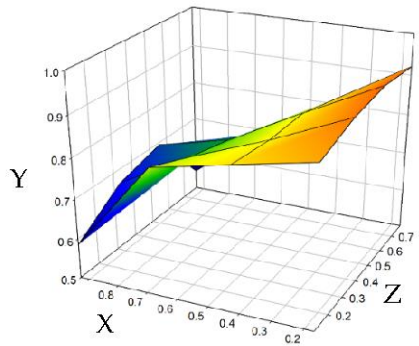
This section provides simulated results in 3D and 2D view to provide in-depth analysis of the outcome of the proposed model for various functions and metrics. Significant samples of the entire experimental results are provided at the latest state of this eFES model development stage. These simulations elaborate on processing features to observe the optimum fitness (i.e., *z* dimension). 3D visuals are selected for better analysis of how the curve moves in space when the learner is optimized in the dimensions. The equation below drives the experimental run for monitoring the *z*-dimension in

correspondence to each of *x*, *y*, and *z*. It should be noted that the results shown are a snapshot of 100+ experimental runs for several data samples of the datasets. The equation shown for each indicates the sampling construct for the analysis being envisioned. Features were included in the experiments from the raw datasets. To improve the generalization of the model, various experiments were performed on standard numbers such as 5, 10, 15, 20 and 40. Clearly, less is more, as we stated earlier, but we leave it up to the model to finally group (FGF) the features that have the highest predictive value for learning and ensuring the maximum fitness and generalization. For each experiment, a miscellaneous dataset was used to improve the generalization ability of the model and underlying algorithms.

Fig. 5.14 shows the 3D variance simulations of the functions. Fig. 5.15 shows the comparison between features that were engineered (Enhanced Feature Engineering (EFE)) and that were not engineered (in blue). It is observed that EFE outperformed the FE. No FE indicates that the experiment took features set as per standard pick and ran the process. EFE indicates the enhanced feature engineering while incorporating mathematical constructs and algorithms, where features were added and removed based on metrics reading and eventually creating an optimum feature set, as engineered by eMLEE.

Fig. 5.16a–d shows the tests on 20-experimental run. It should be noted that as the number of experiments were increased, the classifier learning was improved as per proposed model. The selection of 20 features were based on optimum number of the grouping function (FGF). Clearly, each dataset brings in different number of features. Out of these features, some features are irrelevant, redundant, and outliers. Some features are not known at the beginning of classifier learning. However, we standardized around

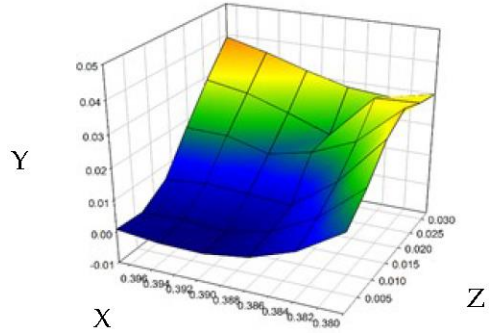
number 20 for experimental purposes. However, it is up to the algorithm to tell the model how many features need to be qualified and then included in the learning process.



$$\frac{1}{l}\sum_{i=1}^{10} d\,(Z\,(x,y,z)\mid \lim_{err>0.2} z_l \in Z$$

x	0.7	0.8	0.2	0.6	0.4
y	0.3	0.5	0.6	0.2	0.9
z	0.2	0.1	0.4	0.5	0.3

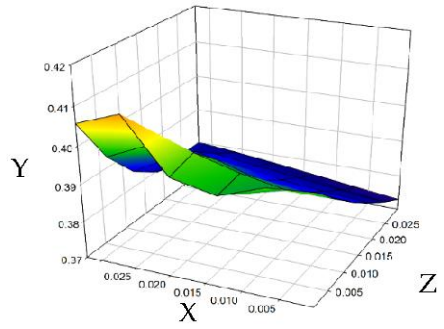
(**a**)



$$\frac{1}{l}\sum_{i=1}^{10} d\,(X\,(x,y,z) + d\,(Y\,(x,y,z) d\,(Z\,(x,y,z)|0.2 < err < 0.8$$

x	0.07	0.06	0.05	0.03	0.05
y	0.01	0.04	0.02	0.01	0.03
z	0.04	0.03	0.04	0.05	0.01

(**b**)



$$\frac{1}{l}\sum_{i=1}^{10} d\,(X\,(x,y,z) + d\,(Y(x,y,z) d(Z(x,y,z)|0$$

x	0.07	0.03	0.04	0.02	0.03
y	0.3	0.5	0.2	0.3	0.04
z	0.03	0.02	0.04	0.01	0.03

(**c**)

It shows that variance in z is minimum on random datasets; (**b**) It shows the variance in all of axis as ideal, as what we wanted to observe; (**c**) It shows the variance in all axis to be real (practical), as what we observed.

Figure 5. 14**.** Shows the 3D variance simulations of the functions

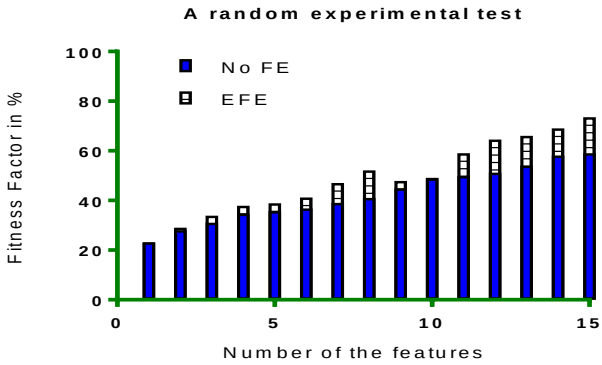


Figure 5. 15**.** A random experiment on 15 features for FE vs. EFE Correlation study for the observed Fitness Factor.

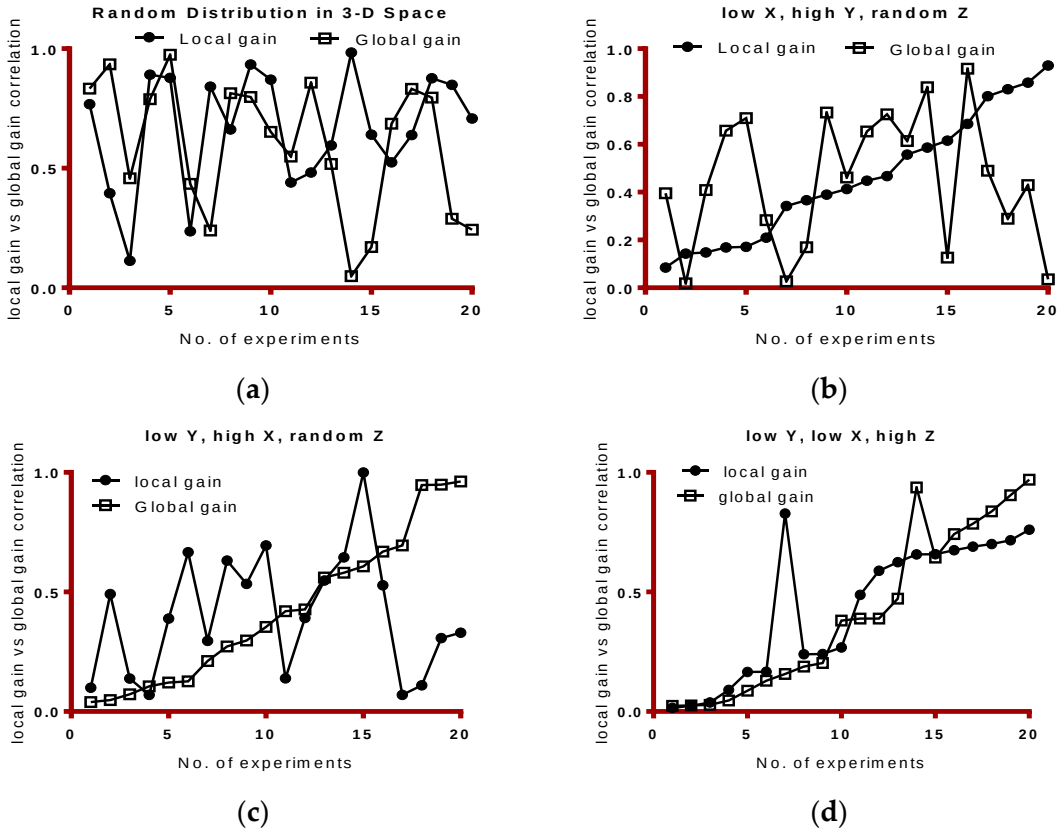
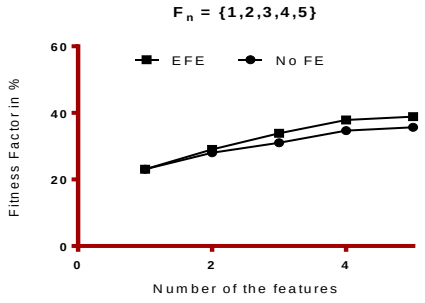


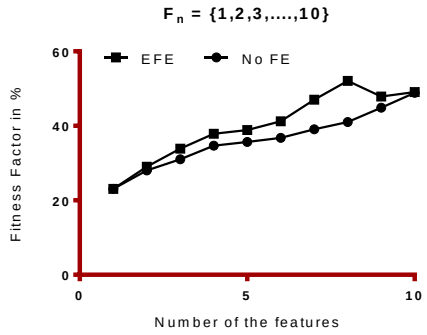
Figure 5. 16**.** Tests on Random Distribution in eFES model

(a) We observe that LG and GG were very random throughout the tests; (b) We observe that LG showed linear correlation (regression) when x (overfitting) was found to be low, and z was kept random in 3D space. GG, as observed was random;

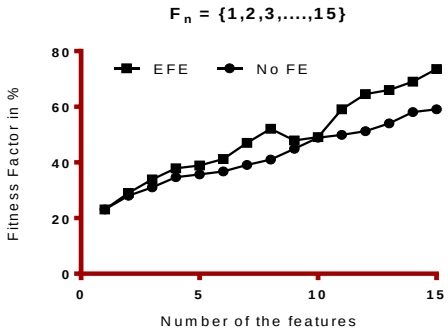
(c) Observations on low y, where we found GG to be close to the linear response; (d) Finally, as the model is optimized (high z), we saw expected and desired linear regression. We observed some unexpected as shown by the peaks, which were suspected to be riding outliers. **Fig. 5.17 a–e** shows the test on (5, 10, 15, 20 and 50) features set. It compares the EFE and FE correlation for Fitness Factor (FF). FF is computed by the eFES algorithms explained earlier. **Fig. 5.17** shows the set of experiments for observation of diverse set of features to study the model's fitness factor. As it is observed that EFE keeps the linearity (stability) of the model. (e) was as special test of various metrics. "Engineered" refers to all the metrics of eFES model incorporated.

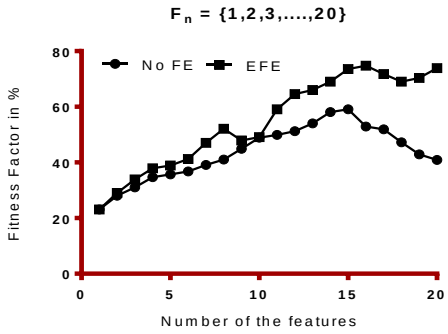


(a)This considers only 5 features to evaluate the fitness function for both EFE and FE.



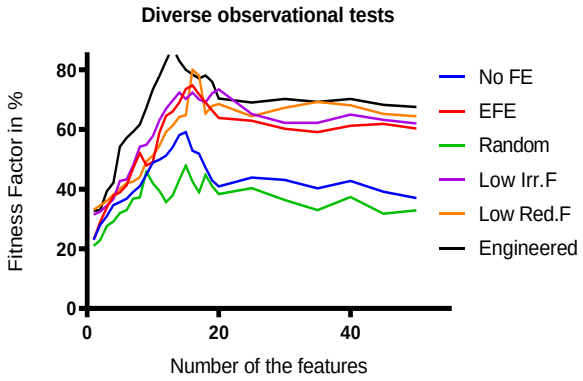
(b) This considers 10 features to evaluate the fitness function for both EFE and FE. Clearly, we observe the improvement in Fitness Function.





(c) With features in higher range of 15, we observe the consistent stability.

(d) However, as expected, we noticed that features up to 20, the maximum range of fitness function is around 80%.



(e) This shows the comparison of the various metrics and read the relevant value of the fitness factor for each study of the metrics as shown by distinct colors.

Figure 5. 17 **.** Set of experiments for observation of diverse set of features to study the model's fitness factor.

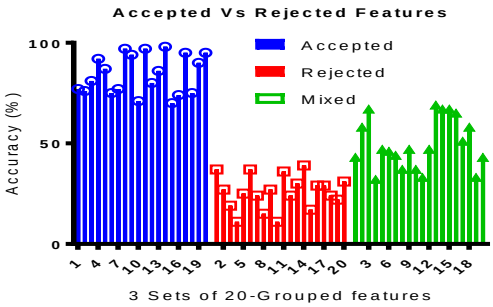
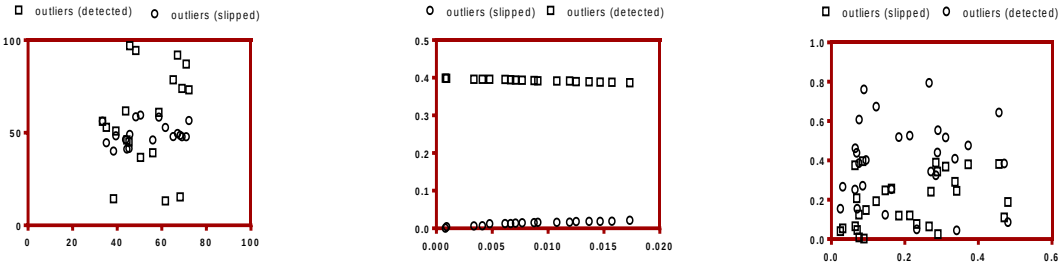


Fig. 5.18. shows the three sets of 20-grouped feature sets. The goal of these experiments was to study the model ability to improve the accuracy for the features (Accepted, Rejected and Mixed) from the given data set.

Figure 5. 18 - Accuracy Validation for Feature Optimization.

Fig 5.19 (a-c) shows the model's ability to detect outliers.



(a) Shows disperse

(b) Improved Detection

(c) Random(uncertain)

Figure 5. 19 Outliers detection experimental results for eFES Unit.

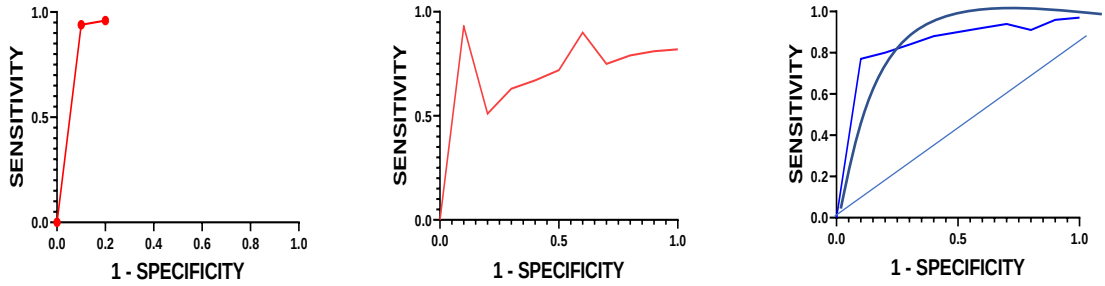
5.5 eMLEE External Testing

5.5.1 Introduction

As a black box eMLEE needs to be tested externally for standard measure such as Accuracy and Errors that basically validate the integrity of any model in the real world. Due to Accuracy paradox known fact, it is not uncommon to use other measure in favor of accuracy, such as Precision, Recall, Sensitivity, Specificity, and F1-Score. In the following section, we present some of the testing we performed on the eMLEE engine as a black box and some of the experiment to elaborate on the flexibility of the engine to customize for metrics tradeoff. Such as if high speed is needed then accuracy may need to be traded off in the lower range.

5.5.2 Performance Metrics Testing

Student Alcohol Consumption, Instances = 1110, Features = 33, Class = Final Grade (division), Sensitivity = TP rate , 1-S = FP Rate



(a) Bad Tuning.

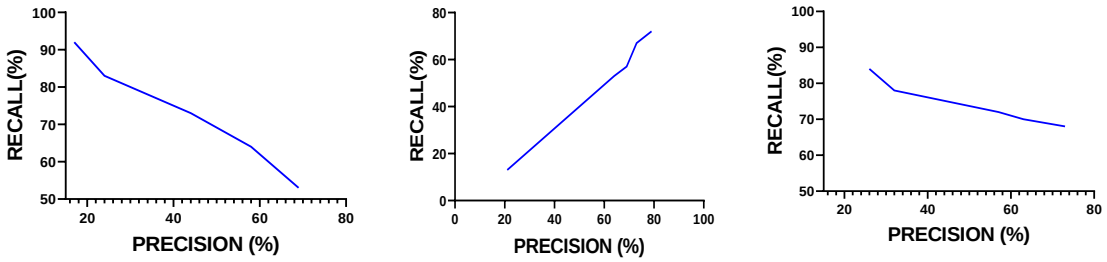
(b) Improved tuning with some undiagnosed errors.

(c) Tuned and desired AUC achieved.

Figure 5. 20 - TP and FP Rate Tests

$$Recall = \frac{tp}{tp+fn}, Precision = \frac{tp}{tp+fp}$$

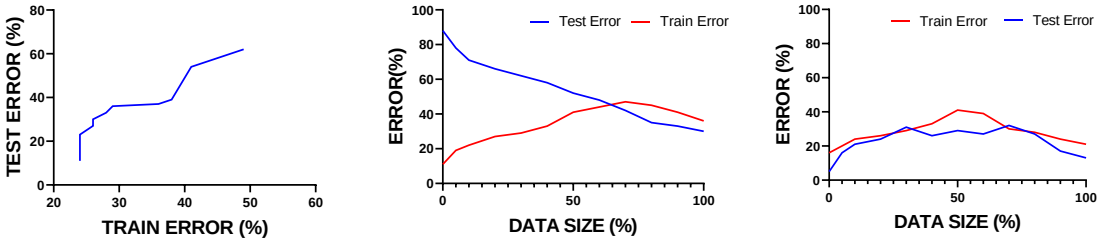
Heart Disease, various sets, Instances = 304, 2512, Features = 14, 7, Class = Yes or No



(a) Typical (b) Extreme error (c) Tuned and Traded-off

Figure 5. 21 - Recall and Precision Test

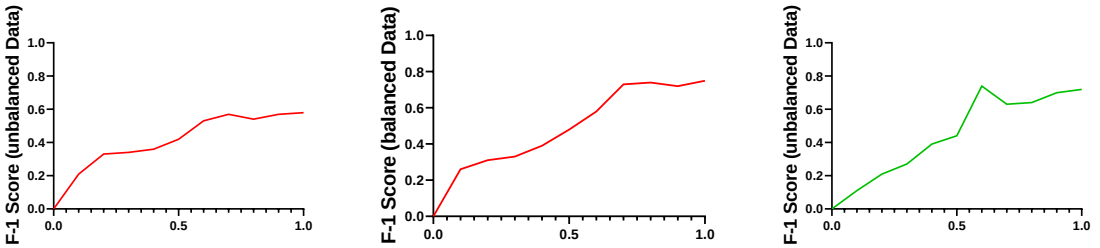
Iris Species, Instances = 150, Features = 6



(a) Correlation study (b) Initial Response (c) Tuned state

Figure 5. 22 - Train Vs Test Error Rates

Pima Indians Diabetes Database, Instances = 768, Features = 9, Class = Yes or No., $H.M = F1 = \frac{2*(Prec*Rec)}{(Prec+Rec)}$



(a) Poor F-1 Score on unbalanced Data (b) Improved F-1 on balanced data (c) Improved and expected Score.

Figure 5. 23 - F-1 Score on unbalanced Data

5.5.3 eMLEE Comparative Testing

Census Data, Features = 13, Instances = 31976, Class: (Salary>50K), (salary>50K)

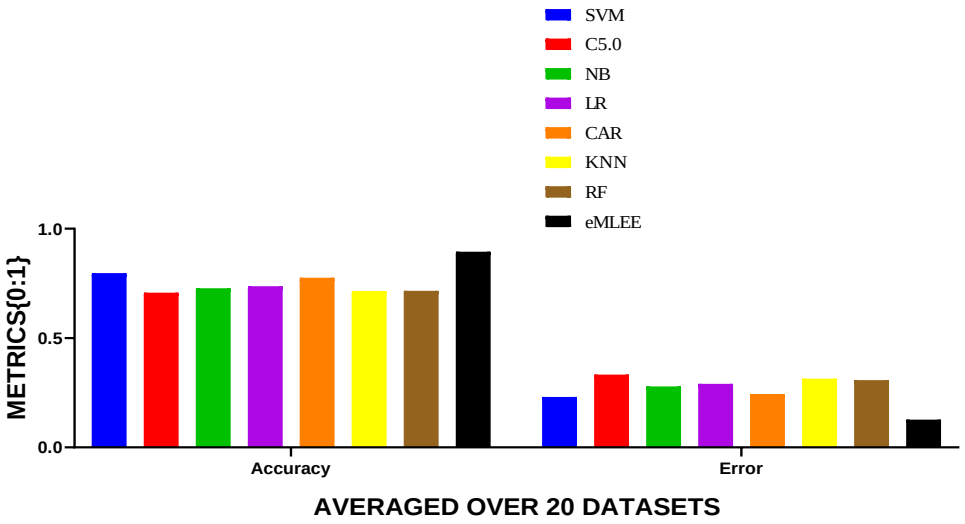


Figure 5. 24 - Accuracy Vs Error on various methods comparison to eMLEE

Census Data, Instances = 31976, Features = 13, **Class**: 1 (Salary>50K), 0 (salary<50K)

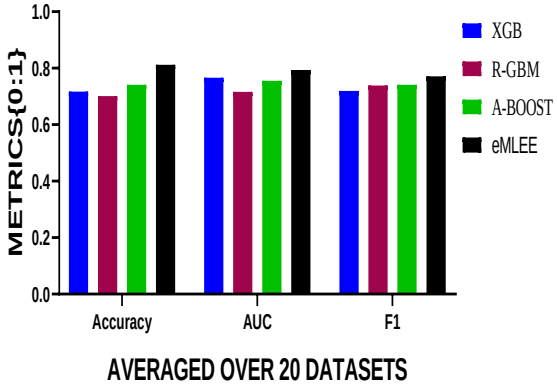


Figure 5. 25 - Accuracy Vs AUC Vs F1 on 20 Datasets for four popular methods and eMLEE

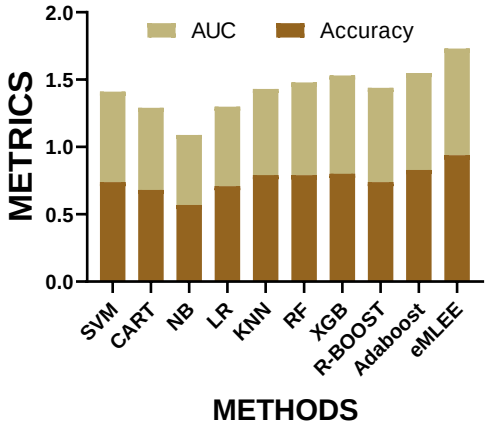


Figure 5. 26 - Shows the aggregation of AUC and Accuracy

Fig 5.27 shows the comparison of eMLEE with standard non-ensemble techniques in the very earlier stages of the development of the engine. Similar results acted as baseline to tune it in competition with state of the art ensemble techniques. The following data sets were used and then results were averaged as shown.

Iris Species, Student Grade Prediction, Credit Card Fraud Detection, and Adult Census Income.

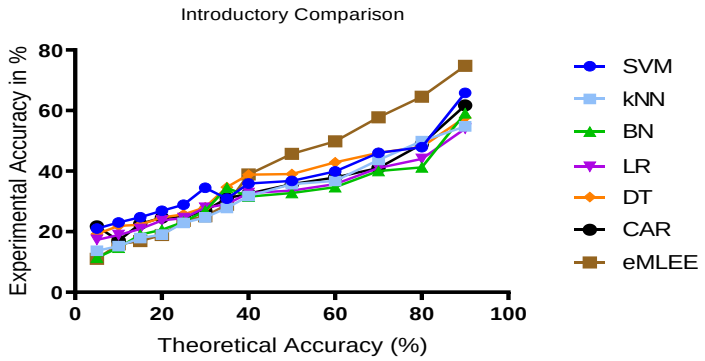
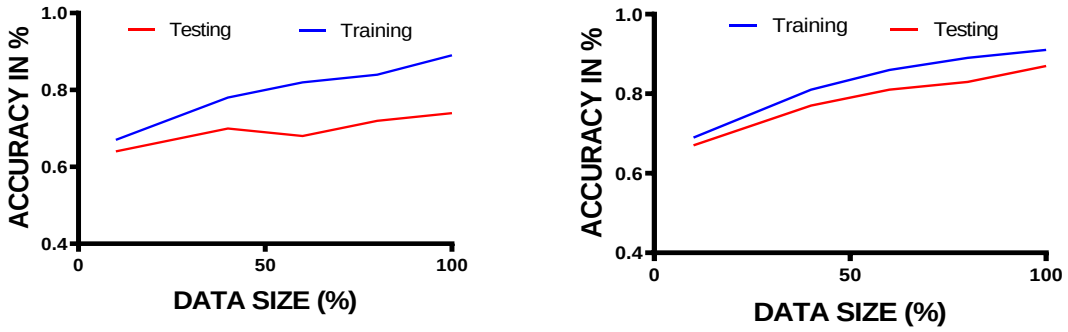


Figure 5. 27– Initial Phases of eMLEE response during development (60 % Development Cycle)

Dataset: Heart Disease, Features: 14 , Instances: 304, Class: Risk Factor (1 for Yes) or (0 for No)



(a) Shows poor testing (XGBOOST) (b) Shows improved testing (EMLEE)

Figure 5. 28– eMLEE Accuracy for Training Vs Testing

5.5.4 Examples

In this section, we provide some examples to support the improvement, we have contributed towards accuracy paradox, using *eWPM* metric.

1. Example of Accuracy Paradox Problem

Dataset: Pima Indians Diabetes Database
No Diabetes = 500 cases (Value '0')
Diabetes = 268 cases (Value '1')
Total = 768 cases
Features = 9
Test Data = 100 cases (90 for '0' and 10 for '1')

PREDICTED \ ACTUAL		
	TP (6)	FP (10)
	FN (12)	TN (72)

$$Accuracy = \frac{6+72}{6+10+12+72} = \mathbf{78\ \%}$$

Observations:

- Predicted 16 to be diabetic, and 84 to be non-diabetic
- It miss-classified 4 out of 10 diabetics, so Accuracy is misleading here
- Accuracy is a good measure, if labelled features are balanced

Improving it using eWPM:

eWPM Metric Pseudocode (Short version)

Split n number of data sets as per LT rules

1. Compute, Avg_acc = $\left[\sum_{i=1}^{L} \frac{(tp_i + tn_i)}{tp_i + fp_i + fn_i + tn_i}\right] / L$ for 'n' times
2. Repeat 1 to 2 for all combination of classifier as per LT error rules and Store in Avg_acc'
3. Run the optimization construct in *LT* unit to compute the optimum points for *x,y,* and *z* and update the row in the table.
4. Compute

$$RoOpFit = \max_{err} < 0.8 \sum_{i,j}^{x,y} (A_{i,j}) \; - \; \min_{err} > 0.2 \sum_{j,i}^{y,x} (A_{j,i})$$

5. Based on RoOpFit, Once LT reports Saturation then STOP.
6. Compute # 2 for Avg_acc' for all new L'

TP (8)	FP (4)
FN (9)	TN (79)

$$Accuracy = \frac{8+79}{8+4+9+79} = \mathbf{87\ \%}$$

Observations:

- Predicted 12 to be diabetic, and 88 to be non-diabetic
- eWPM improved the accuracy measure and it is no more considered misleading.

2. Example of eWPM Vs Standard Accuracy

Dataset: Breast Cancer Wisconsin
Benign(B) = 357
Malignant(M) = 172
Total Cases = 529
Features = 29

Table 5.4 – eWPM based Accuracy test

Iteration	Accuracy	eWPM	Iteration	Accuracy	eWPM
1	0.3412	0.3592	17	0.63	0.7021
2	0.3519	0.371	18	0.6458	0.718
3	0.3711	0.3839	19	0.7099	0.7376
4	0.3914	0.4	20	0.7123	0.751
5	0.4206	0.4206	21	0.679	0.7999
6	0.4413	0.4571	22	0.6823	0.8124
7	0.449	0.4512	23	0.7436	0.8103
8	0.4712	0.4917	24	0.7011	0.8282
9	0.49	0.69	25	0.7267	0.8019
10	0.5204	0.5642	26	0.7228	0.8302
11	0.5321	0.5756	27	0.7112	0.8412
12	0.5911	0.5511	28	0.72	0.8501
13	0.5618	0.5618	29	0.722	0.8507
14	0.5709	0.5709	30	0.7088	0.8594
15	0.5923	0.6274			
16	0.6739	0.6949			

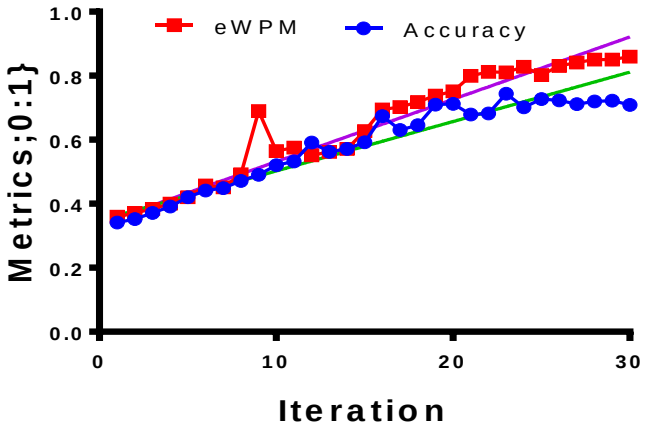


Figure 5. 29 Internal Testing for eMLEE (Integrity and Stability Check).

It is observed that Accuracy influenced by eWPM has shown reliable results when we measure the performance of the proposed model for sampling of 30 iterations.

3. Example 3-F1-Score

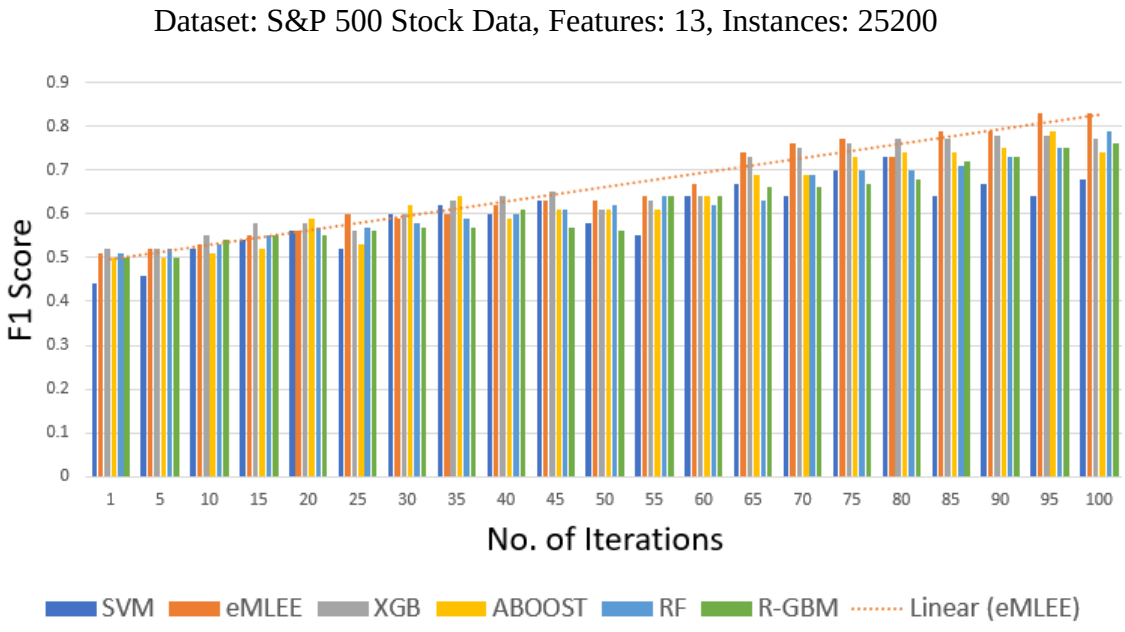


Figure 5. 30 - External Testing for eMLEE (Integrity and Stability Check).

CHAPTER 6 – COMPARATIVE CORROBORATION

6.1 Introduction

In this chapter, we provide the main datasets we used to develop and test our engine internal and external layers. We also provide methods and metrics we used for the relevant validation.

6.1.1 Data and Methods Used

The following three tables list the data sources, methods and metrics.

Table 6.1 – Datasets Used (DS)

1	Breast Cancer Wisconsin Data Set
2	Car Evaluation
3	Iris species
4	Twitter User Gender Classification
5	College Scoreboard
6	Pima Indians Diabetes Database
7	Student Grade Prediction Database
8	Education Statistics
9	Storm Prediction center
10	Fatal Police Shootings
11	2015 Flight Delays and Cancellations
12	Credit Card Fraud Detection
13	Heart disease data set
14	Japan Census data
15	US Mass Shootings
16	Adult Census income
17	1.88 Million US Wildfires
18	S&P 500 stock Data
19	Zika Virus epidemic
20	Student Alcohol Consumption

Table 6.2 - Methods Used

1	Support Vector Machines	SVM
2	Decision Trees(C5.0)	DT
3	Naves Bayes	NB
4	Logistic Regression	LR
5	Decision Trees(CART)	CAR
6	K-Nearest Neighbors	KNN
7	Random Forest	RF
8	XGBoost	XGB
9	Random GBM	R-GBM
10	Adaboost	A-BOOST

Table 6.3 – Metrics Used

1	Accuracy
2	Averaged Error
3	AUC
4	F-Measure
5	Gini Coefficient
6	Correlation Coefficient

6.2 eMLEE Corroboration (Standard Classifiers)

Table 6.4 to 6.9 shows the eMLEE comparisons with other standard methods. It is observed that it outperformed standard methods in most of our testing.

Table 6.4 – Comparison on Accuracy

DS	SVM	DT	NB	LR	CAR	KNN	RF	eMLEE	Winner
1	0.79	0.57	0.67	0.56	0.72	0.62	0.71	0.73	SVM
2	0.66	0.90	0.52	0.90	0.92	0.58	0.69	0.86	MLR
3	0.61	0.71	0.59	0.73	0.83	0.59	0.72	0.80	CAR
4	0.61	0.69	0.80	0.51	0.75	0.55	0.90	0.93	**eMLEE**
5	0.87	0.80	0.75	0.79	0.58	0.81	0.78	0.73	SVM
6	0.83	0.67	0.57	0.83	0.86	0.79	0.59	0.97	**eMLEE**
7	0.90	0.60	0.84	0.88	0.61	0.75	0.58	0.96	**eMLEE**
8	0.52	0.65	0.91	0.64	0.85	0.71	0.93	0.61	RF
9	0.71	0.66	0.69	0.93	0.97	0.89	0.85	0.60	NB
10	0.74	0.50	0.62	0.74	0.79	0.80	0.61	0.87	**eMLEE**
11	0.89	0.52	0.65	0.74	0.88	0.85	0.59	0.95	**eMLEE**
12	0.97	0.81	0.71	0.76	0.96	0.66	0.86	0.90	SVM
13	0.74	0.63	0.85	0.59	0.96	0.64	0.92	0.90	CAR
14	0.51	0.54	0.83	0.70	0.76	0.54	0.88	0.83	RF
15	0.64	0.90	0.81	0.53	0.78	0.72	0.69	0.92	**eMLEE**
16	0.86	0.93	0.83	0.55	0.73	0.69	0.62	0.80	DT
17	0.85	0.87	0.64	0.81	0.67	0.83	0.60	0.78	DT
18	0.95	0.94	0.68	0.86	0.68	0.88	0.84	0.73	SVM
19	0.91	0.94	0.96	0.91	0.89	0.68	0.56	0.98	**eMLEE**
20	0.76	0.90	0.62	0.77	0.51	0.71	0.58	0.84	DT

Table 6.5 – Comparison on Averaged Error

DS	SVM	DT	NB	LR	CAR	KNN	RF	eMLEE	Winner
1	0.19	0.57	0.13	0.29	0.19	0.10	0.57	0.26	KNN
2	0.22	0.20	0.28	0.31	0.48	0.43	0.36	0.43	DT
3	0.51	0.34	0.48	0.41	0.42	0.40	0.24	0.13	**eMLEE**
4	0.08	0.53	0.53	0.58	0.19	0.50	0.39	0.18	SVM
5	0.28	0.14	0.18	0.44	0.49	0.44	0.11	0.28	DT
6	0.25	0.48	0.53	0.10	0.45	0.39	0.49	0.06	**eMLEE**
7	0.27	0.23	0.21	0.29	0.24	0.54	0.36	0.28	NB
8	0.26	0.35	0.48	0.14	0.17	0.48	0.56	0.22	CAR
9	0.14	0.25	0.14	0.15	0.31	0.33	0.32	0.11	**eMLEE**
10	0.30	0.46	0.28	0.49	0.17	0.26	0.25	0.21	CAR
11	0.36	0.41	0.28	0.26	0.12	0.35	0.46	0.11	**eMLEE**
12	0.24	0.44	0.27	0.18	0.12	0.20	0.24	0.21	**CAR**
13	0.23	0.23	0.18	0.28	0.17	0.43	0.26	0.11	eMLEE
14	0.13	0.30	0.34	0.32	0.14	0.18	0.26	0.22	SVM
15	0.08	0.24	0.21	0.29	0.10	0.17	0.15	0.11	SVM
16	0.12	0.27	0.11	0.17	0.15	0.20	0.18	0.04	**eMLEE**
17	0.25	0.36	0.25	0.29	0.31	0.23	0.22	0.13	**eMLEE**

18	0.11	0.28	0.13	0.27	0.25	0.32	0.30	0.19	SVM
19	0.29	0.28	0.38	0.31	0.18	0.18	0.29	0.13	**eMLEE**
20	0.29	0.28	0.17	0.23	0.22	0.14	0.12	0.11	**eMLEE**

Table 6.6 – Comparison on AUC

DS	**SVM**	**DT**	**NB**	**LR**	**CAR**	**KNN**	**RF**	**eMLEE**	**Winner**
1	0.71	0.59	0.91	0.62	0.41	0.73	0.67	0.69	KNN
2	0.63	0.76	0.36	0.77	0.63	0.93	0.89	0.84	KNN
3	0.77	0.85	0.82	0.89	0.85	0.47	0.44	0.92	**eMLEE**
4	0.39	0.93	0.52	0.37	0.45	0.70	0.57	0.78	DT
5	0.47	0.66	0.66	0.34	0.60	0.68	0.34	0.57	KNN
6	0.81	0.47	0.76	0.36	0.49	0.80	0.34	0.50	SVM
7	0.86	0.61	0.59	0.43	0.71	0.58	0.79	0.79	SVM
8	0.61	0.77	0.75	0.53	0.77	0.72	0.55	0.87	**eMLEE**
9	0.71	0.44	0.86	0.66	0.44	0.67	0.73	0.84	**eMLEE**
10	0.67	0.35	0.87	0.37	0.93	0.85	0.35	0.69	KNN
11	0.76	0.70	0.66	0.63	0.86	0.42	0.63	0.84	CAR
12	0.60	0.83	0.87	0.48	0.83	0.38	0.37	0.91	**eMLEE**
13	0.91	0.52	0.66	0.50	0.67	0.68	0.64	0.64	SVM
14	0.57	0.94	0.74	0.61	0.96	0.71	0.41	0.91	CAR
15	0.61	0.73	0.53	0.69	0.73	0.86	0.65	0.80	KNN
16	0.69	0.49	0.73	0.52	0.34	0.67	0.59	0.85	**eMLEE**
17	0.78	0.40	0.81	0.42	0.84	0.78	0.90	0.69	CAR
18	0.65	0.70	0.79	0.75	0.46	0.74	0.45	0.83	**eMLEE**
19	0.69	0.66	0.48	0.56	0.53	0.74	0.63	0.78	**eMLEE**
20	0.54	0.79	0.41	0.67	0.76	0.37	0.40	0.72	DT

Table 6.7 – Comparison on F-Measure

DS	**SVM**	**DT**	**NB**	**LR**	**CAR**	**KNN**	**RF**	**eMLEE**	**Winner**
1	0.56	0.46	0.46	0.70	0.69	0.47	0.62	0.63	LR
2	0.79	0.78	0.72	0.71	0.66	0.38	0.62	0.70	SVM
3	0.76	0.45	0.92	0.42	0.51	0.59	0.72	0.94	**eMLEE**
4	0.79	0.65	0.58	0.73	0.57	0.64	0.60	0.74	SVM
5	0.89	0.74	0.85	0.62	0.83	0.80	0.35	0.76	SVM
6	0.76	0.80	0.68	0.74	0.77	0.77	0.58	0.83	**eMLEE**
7	0.67	0.54	0.66	0.66	0.66	0.65	0.60	0.78	**eMLEE**
8	0.77	0.75	0.63	0.91	0.53	0.42	0.82	0.65	LR
9	0.68	0.56	0.63	0.70	0.71	0.69	0.63	0.79	**eMLEE**
10	0.82	0.80	0.75	0.67	0.78	0.66	0.66	0.72	SVM
11	0.71	0.36	0.37	0.52	0.70	0.75	0.63	0.68	KNN
12	0.84	0.33	0.68	0.72	0.97	0.36	0.71	0.79	SVM
13	0.40	0.73	0.52	0.41	0.83	0.41	0.86	0.88	**eMLEE**
14	0.56	0.58	0.78	0.71	0.80	0.63	0.85	0.87	**eMLEE**
15	0.88	0.44	0.81	0.59	0.89	0.87	0.52	0.76	CAR
16	0.79	0.42	0.36	0.43	0.87	0.47	0.80	0.86	CAR
17	0.76	0.62	0.60	0.70	0.66	0.71	0.63	0.89	**eMLEE**
18	0.65	0.68	0.38	0.68	0.37	0.66	0.50	0.79	**eMLEE**
19	0.64	0.56	0.56	0.78	0.38	0.55	0.50	0.66	LR
20	0.70	0.39	0.59	0.37	0.43	0.76	0.77	0.84	**eMLEE**

Table 6.8 - Comparison on Gini Coefficient

DS	SVM	DT	NB	LR	CAR	KNN	RF	eMLEE	Winner
1	0.76	0.62	0.56	0.86	0.86	0.75	0.66	0.89	**eMLEE**
2	0.95	0.51	0.90	0.75	0.89	0.69	0.88	0.85	SVM
3	0.60	0.77	0.39	0.48	0.80	0.89	0.43	0.81	KNN
4	0.72	0.47	0.43	0.60	0.95	0.97	0.64	0.80	KNN
5	0.92	0.79	0.53	0.70	0.59	0.94	0.93	0.60	SVM
6	0.55	0.39	0.78	0.84	0.53	0.40	0.36	0.96	**eMLEE**
7	0.67	0.90	0.94	0.59	0.83	0.87	0.47	0.93	**eMLEE**
8	0.90	0.44	0.37	0.35	0.79	0.66	0.36	0.84	SVM
9	0.76	0.44	0.58	0.42	0.63	0.79	0.59	0.73	KNN
10	0.61	0.47	0.72	0.52	0.50	0.59	0.54	0.87	**eMLEE**
11	0.85	0.53	0.39	0.96	0.85	0.84	0.86	0.64	SVM
12	0.72	0.94	0.59	0.51	0.69	0.73	0.42	0.65	SVM
13	0.60	0.43	0.39	0.50	0.64	0.61	0.71	0.88	**eMLEE**
14	0.68	0.55	0.68	0.68	0.35	0.53	0.87	0.94	**eMLEE**
15	0.50	0.81	0.49	0.53	0.77	0.41	0.47	0.89	**eMLEE**
16	0.84	0.64	0.52	0.86	0.59	0.51	0.73	0.85	LR
17	0.44	0.79	0.43	0.83	0.80	0.50	0.85	0.90	**eMLEE**
18	0.63	0.50	0.69	0.76	0.96	0.66	0.81	0.97	**eMLEE**
19	0.74	0.76	0.69	0.79	0.49	0.50	0.67	0.70	LR
20	0.80	0.59	0.43	0.97	0.77	0.51	0.56	0.83	LR

Table 6.9 - Comparison on Correlation Coefficient

DS	SVM	DT	NB	LR	CAR	KNN	RF	eMLEE	Winner
1	0.78	0.89	0.54	0.92	0.82	0.92	0.83	0.74	LR
2	0.52	0.81	0.74	0.93	0.65	0.61	0.75	0.98	**eMLEE**
3	0.79	0.63	0.62	0.67	0.68	0.59	0.89	0.76	SVM
4	0.89	0.62	0.67	0.96	0.50	0.48	0.73	0.97	**eMLEE**
5	0.89	0.81	0.63	0.44	0.94	0.47	0.80	0.96	**eMLEE**
6	0.96	0.77	0.94	0.57	0.94	0.78	0.79	0.87	SVM
7	0.67	0.41	0.51	0.45	0.78	0.51	0.88	0.83	RF
8	0.93	0.90	0.44	0.92	0.48	0.65	0.64	0.79	SVM
9	0.87	0.74	0.88	0.77	0.54	0.61	0.74	0.90	**eMLEE**
10	0.80	0.67	0.71	0.67	0.87	0.85	0.97	0.92	**eMLEE**
11	0.92	0.81	0.73	0.56	0.69	0.54	0.49	0.82	SVM
12	0.76	0.88	0.89	0.43	0.76	0.97	0.78	0.92	KNN
13	0.67	0.90	0.47	0.80	0.45	0.91	0.72	0.76	KNN
14	0.72	0.41	0.79	0.82	0.77	0.71	0.51	0.91	**eMLEE**
15	0.56	0.47	0.55	0.90	0.93	0.92	0.74	0.95	**eMLEE**
16	0.81	0.66	0.78	0.79	0.75	0.57	0.94	0.70	SVM
17	0.82	0.43	0.75	0.55	0.48	0.80	0.48	0.87	**eMLEE**
18	0.91	0.83	0.44	0.98	0.60	0.46	0.67	0.96	LR
19	0.82	0.82	0.86	0.99	0.97	0.60	0.49	0.89	LR
20	0.88	0.82	0.77	0.41	0.82	0.62	0.91	0.80	SVM

6.3 eMLEE Corroboration (Ensemble Methods)

Here, we present the testing and comparison of eMLEE as an engine incorporating the most optimized options for utilizing the design of eABT and eFES regulated by LT internals.

Table 6.10 - Comparison with Methods for DS – Census data.

Method	Accuracy	Log.Loss	MSE	AUC
SVM	0.74	0.21	0.25	0.67
CART	0.68	0.32	0.48	0.61
NB	0.57	0.23	0.53	0.52
LR	0.71	0.34	0.18	0.59
KNN	0.79	0.31	0.39	0.64
RF	0.59	0.39	0.49	0.62
XGB	0.8	0.21	0.24	0.73
R-BOOST	0.79	0.25	0.11	0.70
A-BOOST	0.83	0.23	0.17	0.72
eMLEE	0.86	0.19	0.10	0.76

Table 6.11 – Comparison with Methods for Loss Function

DS	XGB	R-GBM	A-BOOST	eMLEE	Winner
1	0.252	0.291	0.231	0.113	eMLEE
2	0.365	0.432	0.191	0.221	A-BOOST
3	0.300	0.391	0.145	0.281	A-BOOST
4	0.571	0.179	0.281	0.273	R-GBM
5	0.429	0.378	0.333	0.298	eMLEE
6	0.472	0.480	0.361	0.320	eMLEE
7	0.360	0.200	0.329	0.287	R-GBM
8	0.267	0.449	0.371	0.380	XGB
9	0.302	0.324	0.253	0.201	eMLEE
10	0.439	0.399	0.329	0.179	eMLEE

Table 6.12 – Comparison with Methods for Cost Function

DS	XGB	R-GBM	A-BOOST	eMLEE	Winner
1	0.238	0.007	0.118	0.386	R-GBM
2	0.128	0.382	0.395	0.297	XGB
3	0.189	0.497	0.227	0.028	eMLEE
4	0.354	0.187	0.097	0.329	A-BOOST
5	0.197	0.285	0.308	0.299	XGB
6	0.119	0.449	0.228	0.018	eMLEE
7	0.349	0.323	0.294	0.115	eMLEE
8	0.448	0.249	0.312	0.394	R-GBM
9	0.352	0.391	0.220	0.134	eMLEE

10	0.139	0.398	0.194	0.229	XGB

Table 6.13 – Comparison on ERROR

DS	XGB	R-GBM	A-BOOST	eMLEE	Winner
1	0.129	0.320	0.173	0.133	XGB
2	0.330	0.294	0.238	0.173	eMLEE
3	0.218	0.329	0.442	0.029	eMLEE
4	0.209	0.139	0.324	0.372	R-GBM
5	0.286	0.219	0.173	0.183	A-BOOST
6	0.170	0.314	0.229	0.119	eMLEE
7	0.009	0.298	0.018	0.128	XGB
8	0.294	0.398	0.091	0.032	eMLEE
9	0.312	0.429	0.219	0.193	eMLEE
10	0.308	0.328	0.023	0.119	A-BOOST

Table 6.14 – Comparison on AUC

DS	XGB	R-GBM	A-BOOST	eMLEE	Winner
1	0.796	0.702	0.761	0.898	eMLEE
2	0.984	0.706	0.803	0.603	XGB
3	0.594	0.555	0.71	0.751	eMLEE
4	0.969	0.65	0.954	0.53	XGB
5	0.702	0.501	0.929	0.682	A-BOOST
6	0.893	0.7	0.626	0.944	eMLEE
7	0.854	0.686	0.932	0.631	A-BOOST
8	0.806	0.865	0.787	0.814	R-GBM
9	0.757	0.768	0.619	0.855	eMLEE
10	0.659	0.649	0.775	0.842	eMLEE
11	0.729	0.856	0.919	0.715	R-GBM
12	0.58	0.723	0.535	0.553	R-GBM
13	0.902	0.777	0.911	0.925	eMLEE
14	0.82	0.814	0.744	0.956	eMLEE
15	0.908	0.706	0.548	0.804	XGB
16	0.687	0.619	0.553	0.813	eMLEE
17	0.628	0.812	0.854	0.579	A-BOOST
18	0.857	0.797	0.665	0.695	XGB
19	0.767	0.631	0.801	0.854	eMLEE
20	0.835	0.811	0.878	0.734	A-BOOST

Table 6.15 – Comparison on ACCURACY

DS	XGB	R-GBM	A-BOOST	eMLEE	Winner
1	0.82	0.797	0.765	0.736	XGB
2	0.772	0.828	0.865	0.896	eMLEE
3	0.663	0.603	0.861	0.745	A-BOOST
4	0.617	0.589	0.702	0.75	eMLEE
5	0.871	0.546	0.51	0.779	XGB
6	0.682	0.764	0.762	0.889	eMLEE
7	0.747	0.771	0.881	0.871	A-BOOST
8	0.742	0.628	0.628	0.891	eMLEE
9	0.722	0.842	0.673	0.871	A-BOOST
10	0.55	0.910	0.889	0.792	R-GBM
11	0.515	0.685	0.704	0.846	eMLEE
12	0.645	0.498	0.66	0.862	eMLEE
13	0.597	0.405	0.678	0.648	A-BOOST
14	0.728	0.612	0.836	0.887	eMLEE
15	0.883	0.701	0.809	0.712	XGB
16	0.845	0.618	0.743	0.714	XGB
17	0.791	0.787	0.731	0.882	eMLEE
18	0.751	0.803	0.794	0.682	R-GBM
19	0.658	0.762	0.76	0.897	eMLEE
20	0.744	0.882	0.567	0.8935	eMLEE

CHAPTER 7 – FINAL REMARKS

We, hereby conclude this thesis with providing the final remarks in the sub-sections below.

7.1 Conclusion

7.1.1 End Goal

The end goal of the scientific work researched and implemented in this thesis was to improve the Supervised Learning (SL) classifier learning process based on improved boosting, bagging, and ensemble of the various existing methods, and improved accuracy-error correlation, generalization, bias, and integrity of the modeling via enhanced feature optimization and improved metrics quantification using *3D* training, parallelism, and visualization approach.

7.1.2 Recap

This thesis reports the latest progress of the proposed engine including mathematical constructs, framework, and algorithms. An enhanced Machine Learning approach has been developed based on the following building blocks (i.e., units):

Centralized Unit as Logical Table (LT) [185] is built using parallelism to regulate and control the classifier learning process. It acts very similar to CPU of the standard

compute. It ensures the blending and tuning of various methods during ensemble process and features engineering are done process and computation effectively. It provides the centralized internal decision unit based on metrics checks during model generations.

i) - Enhanced Algorithm Blend and Tuning (eABT) - a) application of a selected set of supervised learning (*SL*) methods on the experimental dataset and records the measured metrics in a logical table construct, *b)* development of a blend of methods based on in-parallel tuning of model and the classifiers to improve metrics, *c)* development of a logical *3D* cube structure, that governs the algorithms for ensuring optimum fitness for the blend being engineered, *d)* engineering the final engine so it can learn from its mistakes (wrong predictions) and teach itself for picking the right elements and remove the wrong ones during the training process, *e)* and finally validating the proposed model that engine outputs using diverse set of data in the real-world for predictive and prescriptive analytics.

ii) – enhanced Feature Engineering and Selection (eFES) unit [72]– eFES is based on the following building blocks: (a) a features set is processed through standard methods and records the measured metrics; (b) features are weighted based on the learning process where accepted and rejected features are separated using 3D-based training through building Local Gain (LG) and Global Gain (GG) functions; (c) features are then scored so the ML process can evolve into deciding which features need to be accepted or rejected for improved generalization of the model; (d) finally features are evaluated, tested, and the model is completed with feature grouping function (FGF). This thesis reports observation on several hundreds of experiments and then implements 10 experimental approaches to tune the model. The 10th experimental rule was adopted to narrow down (i.e., slice) the result extraction from several hundred runs. The LG and GG functions were built and

optimized in 3D space. The included results show promising outcomes of the proposed scheme of the eFES model. It supports the use of feature sets to further optimize the learning process of ML models for supervised learning. Using the novel approach of Local Error and Global Error bounds of 20% to 80%, we could tune our model more realistically. If the errors were above 80% or below 20%, we flag it to be an invalid fit. This unique approach of engineering a model turns out to be very effective in our experiments and observations, as reported and discussed in this thesis. This model though is based on parallel processing but using high-speed hardware or a Hadoop-based system will help further.

Innovatively, this thesis work invented an enhanced Weighted Performance metric(eWPM) to improve the measurement of the ensemble technique. This research also investigated the improvement on cross validation, as enhanced Cross Validation and Split (eCVS). However, the detailed model building and validation for eWPM and eCVS are outside of this thesis work. These two units are built implicitly in the eMLEE model

7.1.3 Supporting Perceptive

Features (i.e., attributes) in the datasets are often irrelevant and redundant and may have less predictive value. Therefore, we constructed these two functions. *A*) Irrelevant *Irr.F,* and *B*) *Red.F* (Algorithm). The real-world data may have more features and based on this exact fact, we realized the gap to fill with our work. For ML model classifier learning, features play a crucial role when it comes to speed, performance, predictive accuracy, and reliability of the model. Too many features or too few features may overfit or underfit the model. Then the question becomes, what is the optimum (i.e., the right

number) feature set that should be filtered for a ML process, that is where our work comes in. We wanted to have the model decides for itself as it continues to learn with more data. Certain features such as" Gender" may have extreme predictive value (i.e., weight) for building predictive modeling for an academic data from a part of the world where gender bias is high. However, the same feature may not play a significant role when it is included in a set from a domain, where gender bias may not exist. Moreover, we also do not anticipate that based on our thoughts, but we let our model tell us which feature should be included or removed, thus we have two functions, Adder $(+\mathbb{F}(x, y, z))$ and Remover $(-\mathbb{F}(x, y, z)$. Parallel processing and 3D engineering of the features functions greatly improved the FO as we intended to investigate and improved with our work. Future work will further enhance the internals of it.

7.1.4 Highlights and Novelties

a) Parallelized scheme during the blend of the algorithm for the classifier in-time check with the metrics to spot the optimum fitting before algorithm is added or removed in parallel.

b) Error bounds of 20 to 80 % for model to stay in the limits to avoid overfitting and underfitting of the classifier learning process.

c) Specialized gain functions where scoring of each reports back the regulator functions in logical table for maximizing the relevance, reduces the redundancy, improves the fitness, accuracy, and generalization of the model for improved predictive modeling in any datasets.

d) Developing of *Learning From Mistakes* (*LFM*) function to further improve the classifier learning in a unique and enhanced way.

e) A Centralized logical table unit (LT) to regulate the entire learning process in parallel. LT plays a significant role. LT creates parallel process for each element in each run governed by 3D object co-ordinates (x,y and z) and then makes observations in the real time of classifier learning and updates its logical row in the table.

7.1.5 Experimental and Corroboration Approach

Because our research investigated the ensemble of algorithms (that learn based on different classifier curves), we considered a very miscellaneous set of training and testing data to ensure that our blend of algorithms stay in the optimum fitting range for the real-world experiments and analytics. Similarly, because of the feature engineering and tuning, it was authoritative to our work using data with assorted set of features involved.

We also uniquely adopted the following approaches to make our experiments more reliable, easy to interpret, reproduce, and analyze for the model's validation, integrity and evaluation.

3. We conducted several experiments to cover wide range of datasets that helped achieved in-depth training of the model to study various ranges of metrics. We then re-evaluated our math constructs and algorithms to improve fitness. This way, our math constructs governed by our algorithms, ensured the integrity of the model through the lens of real-world data and testing.
4. We also sampled all these experiments and developed a novel approach of 10-experiemntal rule. This way, we could present our outcomes and analysis with improved visualization and interpretation, such as we presented in this research.

We sliced the results based on 10, 20- experimental approach to tune the model and produce simulated results for the analysis. The *LG* and *GG* functions were built and optimized in *3D* space. Finally, it was observed that the parallel tuning and blending approach presented had produced improved results with the potential to generalize on a distinct set of data and problems. Various tabular data were composed form the diverse set of experiments to show the comparison with other techniques.

7.1.6 Errors Encountered

In some of the experimental tests, we came across some invalid outcomes, where we had to re-tune our model. Clearly, every model build-up process contains such issues where more work/investigation is always needed. We have found that such issues are not reflective of any huge inaccuracy in the results or instability of the model. Specially, in our diverse and stress testing, the errors and unexpected behavior and readings were very little as compared to stable and expected results. It should be watched closely with future enhancements and results, so it does not grow and become a real bug. This model is based on supervised learning algorithms.

7.1.7 Engine Anatomy Recap

The proposed engine has two main units as eABT and eFES as explicitly implemented. It has two other units eCVS and eWPM as implicitly implemented. It also introduces a centralized Unit as Logical Table (LT) that is responsible to maintain parallelism for the entire classifier learning. The completed model is constituted of eight

total governing algorithms, various definitions, Lemmas, rules and procedures. The mathematical model in conjunction with underlying governing algorithms builds the proposed model as a main engine known as Enhanced Machine Learning Engineering (eMLEE). We utilized relevant mathematical, statistical and optimization techniques from books and literature to develop the constructs in line governing algorithms that assisted the eMLEE mechanics in the most inner levels.

7.1.8 Closing Comments

The motivation to develop this specialized unit comes from the uniquely thought, experimented, developed, and incorporated parallelism in an enhanced machine learning process with innovative blending and tuning based on Stochastic thinking and Mathematical Modeling. *eMLEE* comes in to addressing "No Free Lunch theorem" problem, feature correlation, and selection improvement. It addresses challenges such as overfit, underfit, bias, predictive errors, and poor generalization. In our experimental tests during the evolution of this research, we felt the necessity of "inline" unit as a centralized part of this engine that governs, regulates, and keeps track of machine learning process on the underlying data. The challenge of trade-off between vital metrics such as complexity, accuracy, speed, etc. becomes also very important and that is where *LT* plays a significant role.

LT creates parallel process for each element in each run governed by 3D object co-ordinates (x,y and z) and then makes observations in the real-time of the classifier learning and then updates its logical row in the table.
This approach is novel to the best of our survey and knowledge.

Innovatively, this thesis work invented an enhanced Weighted Performance metric(eWPM) to improve the measurement of the ensemble technique. This research also

investigated the improvement on cross validation, as enhanced Cross Validation and Split (eCVS).

Finally, eMLEE model was compared with Boosting and Bagging algorithms and relevant data are shown in tables and results findings are presented in the figures.

7.2 Future Work

We have applied the preliminary stages of *eMLEE* to one applied model based on social networking data relevance and unstructured big data[186] known as "Predicting Educational Relevance For an Efficient Classification of Talent (*PERFECT*) algorithm Engine (*PAE)"* [187][188][189][190][191][79] using limited academic and career data[192]. We are working to apply *LT*[73], *eFES*[72], *eABT* (i.e. *eMLEE*) model in its latest form to study/explore/validate further enhancements.

Also our work can lead to pursue research in the directions of unlearning of the machine learning algorithm.

To further improve the current state of the eMLEE and its components (such as reported in this thesis), we will be testing more data specifically from http://www.kaggle.com, www.data.gov, and www.mypersonality.org. We will be developing/testing more algorithms, especially in the domains of unsupervised learning for new insights into feature engineering and selection. Also, *eFES* needs further extensions towards exploring and engineering unknown features that are normally not encountered by the learning process but may have great predictive value. To mature the *eMLEE*, we will be considering many more performance metrics with a much wider range to further

investigate its validation, integrity and stability. We will be looking to build models similar to *PAE* based on *eMLEE* in many other domains such as Healthcare, Stock market, Crime control, Epidemic control, Financial systems, etc.

In the big picture, this thesis can extend into further development of various research problems of the following areas of data science.

1. Reinforcement Learning – Emphasizing on teaching the machine the same way human learns from its mistakes. Experience of human being is basically his memories of the past events, stored and processed by his brain.
2. Artificial Consciousness (AC) – Evolving AI into AC.
3. Machine Learning assisting Deep Learning using AI based real world usages
4. Natural Language Processing
5. Recommender System
6. Mining unknown relationships and structures in noisy and unstructured data
7. Separating Noise from Signal (i.e., Good data vs bad data).
8. Enhanced Decision making (Data to Decisions)
9. Enhanced Talent Correlation with Academics and Careers
10. Genomics data mining – To understand who we are? Why and how we act in the real world. Are we born for specific tasks?

REFERENCES

[1] T. O. Ayodele, "Types of Machine Learning Algorithms," *New Adv. Mach. Learn.*, pp. 19–49, 2010.

[2] P. Domingos, "A few useful things to know about machine learning," *Commun. ACM*, vol. 55, no. 10, p. 78, 2012.

[3] The Economist Intelligence Unit, "The Deciding Factor: Big Data & Decision Making," *Capgemini*, pp. 1–24, 2012.

[4] C. Elkan, *Predictive analytics and data mining*. 2010.

[5] F. Pianesi, "Searching for personality," *IEEE Signal Process. Mag.*, vol. 30, no. 1, pp. 146–158, 2013.

[6] A. Vinciarelli and G. Mohammadi, "A survey of personality computing," *IEEE Trans. Affect. Comput.*, vol. 5, no. 3, pp. 273–291, 2014.

[7] P. Branco, L. Torgo, and R. P. Ribeiro, "A Survey of Predictive Modeling on Imbalanced Domains," *ACM Comput. Surv.*, vol. 49, no. 2, pp. 1–50, 2016.

[8] K. V Kanimozhi and M. Venkatesan, "Unstructured Data Analysis-A Survey," vol. 4, no. 3, pp. 223–225, 2015.

[9] S. Feldman, J. Hanover, C. Burghard, and D. Schubmehl, "Unlocking the Power of Unstructured Data," *IDC Heal. Insights*, vol. June, no. June 2012, 2012.

[10] Y.-C. J. Wu, W.-H. Chang, and C.-H. Yuan, "Do Facebook profile pictures reflect user's personality?," *Comput. Human Behav.*, Dec. 2014.

[11] M. Arias, A. Arratia, and R. Xuriguera, “Forecasting with twitter data,” *ACM Trans. Intell. Syst. Technol.*, vol. 5, no. 1, pp. 1–24, 2013.

[12] M. Russell, *Mining the Social Web: Data Mining Facebook, Twitter, LinkedIn, Google+, GitHub, and More*. 2011.

[13] S. Ben-David and S. Shalev-Shwartz, *Understanding Machine Learning: From Theory to Algorithms*. 2014.

[14] R. Bott, *Active Learning Challenge*, vol. 6, no. 1. 2014.

[15] A. L’Heureux, K. Grolinger, H. F. Elyamany, and M. A. M. Capretz, “Machine Learning With Big Data: Challenges and Approaches,” *IEEE Access*, vol. 5, pp. 7776–7797, 2017.

[16] Mathworks, “Machine Learning Challenges : Choosing the Best Model and Avoiding Overfitting,” pp. 1–8, 2016.

[17] K. Fussman, “The Paradox of,” no. April, 2010.

[18] I. Guyon and A. Elisseeff, “An Introduction to Variable and Feature Selection,” *J. Mach. Learn. Res.*, vol. 3, no. 3, pp. 1157–1182, 2003.

[19] A. Globerson and N. Tishby, “Sufficient Dimensionality Reduction,” *J. Mach. Learn. Res.*, vol. 3, no. 7–8, pp. 1307–1331, 2003.

[20] Yu, Lei, and H. Liu, “Efficient feature selection via analysis of relevance and redundancy,” *J. Mach. Learn. Res.*, vol. 5, pp. 1205–1224, 2004.

[21] Z. Li and W. Gu, “A redundancy-removing feature selection algorithm for nominal data,” *PeerJ Comput. Sci.*, vol. 1, p. e24, 2015.

[22] M. Ramaswami and R. Bhaskaran, “A Study on Feature Selection Techniques in Educational Data Mining,” *J. Comput.*, vol. 1, no. 1, pp. 7–11, 2009.

[23] A. L. Blum and P. Langley, “Selection of relevant features and examples in machine learning,” *Artif. Intell.*, vol. 97, no. 1–2, pp. 245–271, 1997.

[24] “Supervised Learning,” pp. 1–3.

[25] J. Li *et al.*, “Feature Selection: A Data Perspective,” pp. 1–73, 2016.

[26] P. Dayan, “Unsupervised learning,” *MIT Encycl. Cogn. Sci.*, pp. 1–7, 2009.

[27] K. Chai, H. T. Hn, and H. L. Cheiu, “Naive-Bayes Classification Algorithm,” *Bayesian Online Classif. Text Classif. Filter.*, pp. 97–104, 2002.

[28] D. Tuia, M. Volpi, L. Copa, M. Kanevski, and J. Munoz-Mari, “A Survey of Active Learning Algorithms for Supervised Remote Sensing Image Classification,” *IEEE J. Sel. Top. Signal Process.*, vol. 5, no. 3, pp. 606–617, 2011.

[29] K. J. Danjuma, “Performance Evaluation of Machine Learning Algorithms in Post-operative Life Expectancy in the Lung Cancer Patients,” *J. Comput. Sci. Issues*, 2015.

[30] J. Huang, “Performance measures of machine learning,” *Cornel Univ.*, pp. 1–32, 2006.

[31] J. S. Akosa, “Predictive Accuracy : A Misleading Performance Measure for Highly Imbalanced Data,” *SAS Glob. Forum*, vol. 942, pp. 1–12, 2017.

[32] Y. Yao, Z. Xiao, B. Wang, H. Zheng, B. Y. Zhao, and B. Viswanath, “Complexity vs. Performance: Empirical Analysis of Machine Learning as a Service,” *2017 Internet Meas. Conf.*, no. 119, pp. 384–397, 2017.

[33] R. Caruana, N. Karampatziakis, and A. Yessenalina, “An empirical evaluation of supervised learning in high dimensions,” *Proc. 25th Int. Conf. Mach. Learn.*, pp. 96–103, 2008.

[34] P.-N. Tan, M. Steinbach, and V. Kumar, “Classification : Basic Concepts , Decision Trees , and,” *Introd. to Data Min.*, vol. 67, no. 17, pp. 145–205, 2006.

[35] K. M. Leung, “k-Nearest Neighbor Algorithm for Classification,” 2007.

[36] M. A. Hearst, S. T. Dumais, E. Osman, J. Platt, and B. Scholkopf, “Support vector machines,” *IEEE Intell. Syst.*, vol. 13, pp. 18–28, 1998.

[37] R. G. Brereton and G. R. Lloyd, “Support vector machines for classification and regression.,” *Analyst*, vol. 135, no. 2, pp. 230–267, 2010.

[38] M. Eirinaki and M. Vazirgiannis, “Web site personalization based on link analysis and navigational patterns,” *ACM Trans. Internet Technol.*, vol. 7, no. 4, p. 21–es, 2007.

[39] J. Higgins, “Introduction to Mulitple Regression,” *Radic. Stat.*, pp. 1–15, 2005.

[40] S. J. Sheather, “Logistic Regression,” *A Mod. Approach to Regres. with R*, pp. 263–303, 2009.

[41] T. Hazan, A. G. Schwing, and R. Urtasun, “Blending Learning and Inference in Conditional Random Fields,” *J. Mach. Learn. Res.*, vol. 17, pp. 1–22, 2016.

[42] R. E. Schapire, Y. Freund, P. Bartlett, and S. Lee, “Boosting the Margin: a New Explanation for the Effectiveness of Voting Methods,” *Ann. Stat.*, vol. 26, no. 5, p. 1686, 1998.

[43] “Fundamental Issues in Machine Learning,” pp. 1–11.

[44] P. Alves, S. Liu, D. Wang, and M. Gerstein, “Multiple-Swarm Ensembles: Improving the Predictive Power and Robustness of Predictive Models and Its Use in Computational Biology,” *IEEE/ACM Trans. Comput. Biol. Bioinforma.*, vol. 5963, no. c, pp. 1–1, 2017.

[45] A. Figueroa and J. Atkinson, “Ensembling Classifiers for Detecting User Intentions behind Web Queries,” *IEEE Internet Comput.*, vol. 20, no. 2, pp. 8–16, 2016.

[46] S. Paisitkriangkrai, C. Shen, and A. Van Den Hengel, “Pedestrian Detection with Spatially Pooled Features and Structured Ensemble Learning,” *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 38, no. 6, pp. 1243–1257, 2016.

[47] H. Liu and H. Motoda, *Feature selection for knowledge discovery and data mining*. 1998.

[48] G. Forman, “Feature Selection for Text Classification,” *Comput. Methods Featur. Sel.*, vol. 16, pp. 257–274, 2007.

[49] M. S. Nixon and A. S. Aguado, *Feature Extraction & Image Processing for Computer Vision*. 2012.

[50] V. Kumar, “Feature Selection: A literature Review,” *Smart Comput. Rev.*, vol. 4, no. 3, 2014.

[51] J. Li *et al.*, “Feature Selection: A Data Perspective,” vol. 50, no. 6, 2016.

[52] I. Guyon and A. Elisseeff, “Feature Extraction, Foundations and Applications: An introduction to feature extraction,” *Stud. Fuzziness Soft Comput.*, vol. 207, pp. 1–25, 2006.

[53] E. LeDell, “Scalable Ensemble Learning and Computationally Efficient Variance Estimation,” *Thesis*, vol. 53, no. 9, pp. 1689–1699, 2013.

[54] E. García and F. Lozano, “Boosting Support Vector Machines,” *Proc. Int. Conf. Mach. Learn. Data Min.*, pp. 153–167, 2007.

[55] D. Achlioptas, F. McSherry, and B. Schölkopf, “Sampling Techniques for Kernel Methods,” *Adv. Neural Inf. Process. Syst.*, pp. 335–342, 2002.

[56] M. Kirk, *Thoughtful Machine Learning*. O'Reilly Media, 2015.

[57] L. D. Miller and L. K. Soh, "Cluster-based boosting," *IEEE Trans. Knowl. Data Eng.*, vol. 27, no. 6, pp. 1491–1504, 2015.

[58] N. Hatami, "Some Proposals for Combining Ensemble Classifiers," *Computer (Long. Beach. Calif).*, no. March, 2012.

[59] J. Zhao, L. Wang, R. Cabral, and F. De Torre, "Feature and Region Selection for Visual Learning," vol. 25, no. 3, pp. 1084–1094, 2016.

[60] Kaggle, "Feature engineering," pp. 1–11, 2010.

[61] U. Khurana *et al.*, "Automated Feature Engineering for Predictive Modeling ' Kernel- Based Feature Extraction For Collaborative Filtering ' Data Science Workflow."

[62] L. Ohno-machado, "Cross-validation and Bootstrap Ensembles, Bagging, Boosting," *Spring*, 2005.

[63] T. T. Wong and N. Y. Yang, "Dependency Analysis of Accuracy Estimates in k-Fold Cross Validation," *IEEE Trans. Knowl. Data Eng.*, vol. 29, no. 11, pp. 2417–2427, 2017.

[64] N. Japkowicz, "Why Question Machine Learning Evaluation Methods ? (An illustrative review of the shortcomings of current methods)," *AAAI-2006 Work. Eval. Methods Mach. Learn.*, p. 6, 2006.

[65] M. Hossin and M. N. Sulaiman, "Review on Evaluation Metrics for Data Classification Evaluations," *Int. J. Data Min. Knowl. Manag. Process*, vol. 5, no. 2, pp. 1–11, 2015.

[66] S. García, A. Fernández, J. Luengo, and F. Herrera, "A study of statistical techniques

and performance measures for genetics-based machine learning: Accuracy and interpretability," *Soft Comput.*, vol. 13, no. 10, pp. 959–977, 2009.

[67] H. Xia and S. C. H. Hoi, "MKBoost: A framework of multiple kernel boosting," *IEEE Trans. Knowl. Data Eng.*, vol. 25, no. 7, pp. 1574–1586, 2013.

[68] C. Shen, G. Lin, and A. Van Den Hengel, "StructBoost: Boosting Methods for Predicting Structured Output Variables," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 36, no. 10, pp. 2089–2103, 2014.

[69] Y.-P. Zhao, Y.-T. Pan, F.-Q. Song, L. Sun, and T.-H. Chen, "Feature selection of generalized extreme learning machine for regression problems," *Neurocomputing*, vol. 275, pp. 2810–2823, 2017.

[70] M. L. Faq, "Model evaluation , model selection , and algorithm selection in machine learning Performance Estimation : Generalization Performance Vs . Model Selection," no. January, pp. 1–13, 2016.

[71] N.Bahl, "Optimizing Performance Measures in Classification," no. May, 2017.

[72] M. F. Uddin, J. Lee, S. Rizvi, and S. Hamada, "Proposing enhanced feature engineering and a selection model for machine learning processes," *Appl. Sci.*, vol. 8, no. 4, 2018.

[73] M. F. Uddin, S. Rizvi, and A. Razaque, "Proposing Logical Table Constructs for Enhanced Machine Learning Process," *IEEE Access*, vol. PP, no. c, pp. 1–1, 2018.

[74] J. Batiz-Benet, Q. Slack, M. Sparks, and A. Yahya, "Parallelizing Machine Learning Algorithms," pp. 1–5, 2013.

[75] L. Li, Y. Lin, N. Zheng, and F. Wang, "Parallel Learning : a Perspective and a Framework," vol. 4, no. 3, pp. 389–395, 2017.

[76] K. Siddique, Z. Akhtar, H. Lee, W. Kim, and Y. Kim, “Toward Bulk Synchronous Parallel-Based Machine Learning Techniques for Anomaly Detection in High-Speed Big Data Networks,” *Symmetry (Basel).*, vol. 9, no. 9, p. 197, 2017.

[77] M. Architectures, W. Li, H. Fu, Y. You, L. Yu, and J. Fang, “Parallel Multiclass Support Vector Machine for Remote Sensing Data Classification on Multicore,” vol. 10, no. 10, pp. 1–12, 2017.

[78] M. F. Uddin and J. Lee, “Proposing stochastic probability-based math model and algorithms utilizing social networking and academic data for good fit students prediction,” *Soc. Netw. Anal. Min.*, vol. 7, no. 1, 2017.

[79] M. F. Uddin and J. Lee, “Predicting good fit students by correlating relevant personality traits with academic/career data,” *Proc. 2016 IEEE/ACM Int. Conf. Adv. Soc. Networks Anal. Mining, ASONAM 2016*, pp. 968–975, 2016.

[80] T. G. Dietterich, “Ensemble Methods in Machine Learning,” *Mult. Classif. Syst.*, vol. 1857, pp. 1–15, 2000.

[81] C. Silva, U. Lotrič, B. Ribeiro, and A. Dobnikar, “Distributed text classification with an ensemble kernel-based learning approach,” *IEEE Trans. Syst. Man Cybern. Part C Appl. Rev.*, vol. 40, no. 3, pp. 287–297, 2010.

[82] P. K. Mallapragada, R. Jin, A. K. Jain, and Y. Liu, “SemiBoost: Boosting for semi-supervised learning,” *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 31, no. 11, pp. 2000–2014, 2009.

[83] F. Bao, Y. Deng, Y. Zhao, J. Suo, and Q. Dai, “Bosco: Boosting Corrections for Genome-Wide Association Studies with Imbalanced Samples,” *IEEE Trans. Nanobioscience*, vol. 16, no. 1, pp. 69–77, 2017.

[84] X. Liu, “Ensemble Transfer Learning Algorithm,” pp. 2389–2396, 2018.

[85] Z. H. Zhou and Y. Yu, “Ensembling local learners through multimodal perturbation,” *IEEE Trans. Syst. Man, Cybern. Part B Cybern.*, vol. 35, no. 4, pp. 725–735, 2005.

[86] P. Sun and X. Yao, “Sparse approximation through boosting for learning large scale kernel machines,” *IEEE Trans. Neural Networks*, vol. 21, no. 6, pp. 883–894, 2010.

[87] J. Bergstra, N. Pinto, and D. Cox, “Machine learning for predictive auto-tuning with boosted regression trees,” *2012 Innov. Parallel Comput. InPar 2012*, 2012.

[88] N. Asadi, J. Lin, and A. P. De Vries, “Runtime Optimizations for Tree-Based Machine Learning Models,” *IEEE Trans. Knowl. Data Eng.*, vol. 26, no. 9, pp. 2281–2292, 2014.

[89] A. Samat, P. J. Du, S. C. Liu, J. Li, and L. Cheng, “(ELMs)-L-2: Ensemble Extreme Learning Machines for Hyperspectral Image Classification,” *Ieee J. Sel. Top. Appl. Earth Obs. Remote Sens.*, vol. 7, no. 4, pp. 1060–1069, 2014.

[90] M. A. Bencherif, Y. Bazi, A. Guessoum, N. Alajlan, F. Melgani, and H. Alhichri, “Fusion of extreme learning machine and graph-based optimization methods for active classification of remote sensing images,” *IEEE Geosci. Remote Sens. Lett.*, vol. 12, no. 3, pp. 527–531, 2015.

[91] N. Tandon, A. S. Varde, and G. De Melo, “Commonsense Knowledge in Machine Intelligence,” pp. 49–52.

[92] Á. B. Hernández, M. S. Perez, S. Gupta, and V. Muntés-Mulero, “Using machine learning to optimize parallelism in big data applications,” *Futur. Gener. Comput. Syst.*, 2017.

[93] Q. Dai and G. Song, “A novel Supervised Competitive Learning algorithm,” *Neurocomputing*, vol. 191, pp. 356–362, 2016.

[94] S. García, J. Luengo, J. A. Sáez, V. López, and F. Herrera, “A survey of discretization techniques: Taxonomy and empirical analysis in supervised learning,” *IEEE Trans. Knowl. Data Eng.*, vol. 25, no. 4, pp. 734–750, 2013.

[95] Y. Wang, V. Chattaraman, H. Kim, and G. Deshpande, “Predicting Purchase Decisions Based on Spatio-Temporal Functional MRI Features Using Machine Learning,” *IEEE Trans. Auton. Ment. Dev.*, vol. 7, no. 3, pp. 248–255, 2015.

[96] B. Liu, H. Aliakbarian, Z. Ma, G. A. E. Vandenbosch, G. Gielen, and P. Excell, “An Efficient Method for Antenna Design Optimization Based on Evolutionary Computation and Machine Learning Techniques,” *IEEE Trans. Antennas Propag.*, vol. 62, no. 1, pp. 7–18, 2014.

[97] Z. Yu *et al.*, “Adaptive Ensembling of Semi-Supervised Clustering Solutions,” *IEEE Trans. Knowl. Data Eng.*, vol. 29, no. 8, pp. 1577–1590, 2017.

[98] D. Xiao-jian, L. Yuan, Z. Zhi-feng, and X. xin, “Optimization extreme learning machine with ν regularization,” *Neurocomputing*, vol. 261, pp. 11–19, 2017.

[99] O. D. Lara and M. A. Labrador, “A Survey on Human Activity Recognition using Wearable Sensors,” *IEEE Commun. Surv. Tutorials*, vol. 15, no. 3, pp. 1192–1209, 2013.

[100] J. R. Vergara and P. A. Estévez, “A Review of Feature Selection Methods Based on Mutual Information,” 2015.

[101] Y. Mohsenzadeh, H. Sheikhzadeh, A. M. Reza, N. Bathaee, and M. M. Kalayeh, “The relevance sample-feature machine: A sparse bayesian learning approach to

joint feature-sample selection," *IEEE Trans. Cybern.*, vol. 43, no. 6, pp. 2241–2254, 2013.

[102] X. Ma, H. Wang, B. Xue, M. Zhou, B. Ji, and Y. Li, "Depth-based human fall detection via shape features and improved extreme learning machine," *IEEE J. Biomed. Heal. Informatics*, vol. 18, no. 6, pp. 1915–1922, 2014.

[103] D. Lam and D. Wunsch, "Unsupervised Feature Learning Classification with Radial Basis Function Extreme Learning Machine Using Graphic Processors," *IEEE Trans. Cybern.*, vol. 47, no. 1, pp. 224–231, 2017.

[104] Z. Han, Z. Liu, J. Han, C. M. Vong, S. Bu, and X. Li, "Unsupervised 3D Local Feature Learning by Circle Convolutional Restricted Boltzmann Machine," *IEEE Trans. Image Process.*, vol. 25, no. 11, pp. 5331–5344, 2016.

[105] Y. Zeng, X. Xu, D. Shen, Y. Fang, and Z. Xiao, "Traffic Sign Recognition Using Kernel Extreme Learning Machines With Deep Perceptual Features," *IEEE Trans. Intell. Transp. Syst.*, vol. 18, no. 6, pp. 1–7, 2016.

[106] K. Zhang *et al.*, "Machine Learning-Based Temperature Prediction for Runtime Thermal Management Across System Components," *IEEE Trans. Parallel Distrib. Syst.*, vol. 29, no. 2, pp. 405–419, 2018.

[107] J. Wang, G. Wang, and M. Zhou, "Bimodal Vein Data Mining via Cross-Selected-Domain Knowledge Transfer," vol. 13, no. 3, pp. 733–744, 2018.

[108] M. Liu, C. Xu, Y. Luo, C. Xua, Y. Wen, and D. Tao, "Cost-Sensitive Feature Selection by Optimizing F-measures," *IEEE Trans. Image Process.*, vol. 27, no. 3, pp. 1323–1335, 2017.

[109] A. Abbas, S. Member, I. F. Siddiqui, S. Uk, and J. I. N. Lee, "Multi-Objective

Optimum Solutions for IoT-Based Feature Models of Software Product Line," vol. 6, 2018.

[110] J. He and W. W. Chu, "A Social Network-Based Recommender System (SNRS)," *Data Min. Soc. Netw. Data*, vol. 12, pp. 47–74, 2010.

[111] N. M. Eklaspur and A. S. Pashupatimath, "A Friend Recommender System for Social Networks by Life Style Extraction Using Probabilistic Method - ' Friendtome ,'" vol. 3, no. 3, pp. 95–101, 2015.

[112] F. Khoshnood and M. Mahdavi, "Designing a recommender system based on social network and location based services," *Int. J. Manag. Inf. Technol.*, vol. 4, no. 4, pp. 41–47, 2012.

[113] J. Lee, K. Lee, and J. Kim, "Personalized Academic Research Paper Recommendation System," *arXiv Prepr. arXiv1304.5457*, pp. 1–8, 2013.

[114] C. V. Sacín, J. B. Agapito, L. Shafti, and A. Ortigosa, "Recommendation in Higher Education Using Data Mining Techniques," *Proc. 2nd Int. Conf. Educ. Data Min.*, pp. 191–199, 2009.

[115] N. Thai-Nghe, L. Drumond, A. Krohn-Grimberghe, and L. Schmidt-Thieme, "Recommender system for predicting student performance," *Procedia Comput. Sci.*, vol. 1, no. 2, pp. 2811–2819, 2010.

[116] R. Wald, T. Khoshgoftaar, and C. Sumner, "Machine prediction of personality from Facebook profiles," *Proc. 2012 IEEE 13th Int. Conf. Inf. Reuse Integr. IRI 2012*, pp. 109–115, 2012.

[117] J. Golbeck, C. Robles, M. Edmondson, and K. Turner, "Predicting personality from twitter," *Proc. - 2011 IEEE Int. Conf. Privacy, Secur. Risk Trust IEEE Int. Conf.*

Soc. Comput. PASSAT/SocialCom 2011, pp. 149–156, 2011.

[118] E. L. Dey and A. W. Astin, "Predicting College Student Retention," *Comp. Natl. Data from 1982 Freshm. Cl.*, p. 44, 2011.

[119] L. Cen, D. Ruta, and J. Ng, "Big Education : Opportunities for Big Data Analytics," pp. 502–506, 2015.

[120] R. Alkhasawneh and R. Hobson, "Modeling student retention in science and engineering disciplines using neural networks," *2011 IEEE Glob. Eng. Educ. Conf. EDUCON 2011*, pp. 660–663, 2011.

[121] R. J. Haddad and Y. Kalaani, "Can computational thinking predict academic performance?," *ISEC 2015 - 5th IEEE Integr. STEM Educ. Conf.*, pp. 225–229, 2015.

[122] F. Sarker, T. Tiropanis, and H. C. Davis, "Linked data, data mining and external open data for better prediction of at-risk students," *Proc. - 2014 Int. Conf. Control. Decis. Inf. Technol. CoDIT 2014*, pp. 652–657, 2014.

[123] A. Sharabiani, F. Karim, M. Atanasov, and H. Darabi, "An Enhanced Bayesian Network Model for Prediction of Students' Academic Performance in Engineering Programs," *Proc. IEEE Glob. Eng. Educ. Conf.*, no. April, pp. 832–837, 2014.

[124] A. Slim, G. L. Heileman, J. Kozlick, and C. T. Abdallah, "Employing markov networks on curriculum graphs to predict student performance," *Proc. - 2014 13th Int. Conf. Mach. Learn. Appl. ICMLA 2014*, pp. 415–418, 2014.

[125] A. Merceron and K. Yacef, "Educational data mining: A case study," *Artif. Intell. Educ. Support. Learn. through Intell. Soc. Inf. Technol.*, pp. 467–474, 2005.

[126] G. W. . Dekker, M. . Pechenizkiy, and J. M. . Vleeshouwers, "Predicting students

drop out: A case study," *EDM'09 - Educ. Data Min. 2009 2nd Int. Conf. Educ. Data Min.*, pp. 41–50, 2009.

[127] J. Sheard, J. Ceddia, J. Hurst, and J. Tuovinen, "Inferring student learning behaviour from website interactions: A usage analysis," *Educ. Inf. Technol.*, vol. 8, no. 2002, pp. 245–266, 2003.

[128] A. El-Halees, "Mining Students Data To Analyze Learning Behavior : a Case Study Educational Systems," *Work*, 2008.

[129] M. Goga, S. Kuyoro, and N. Goga, "A Recommender for Improving the Student Academic Performance," *Procedia - Soc. Behav. Sci.*, vol. 180, no. November 2014, pp. 1481–1488, 2015.

[130] J. K. J. Kalpana and K. Venkatalakshmi, "Intellectual Performance Analysis of Students' by using Data Mining Techniques," vol. 3, no. 3, pp. 1922–1929, 2014.

[131] A. A. Al-shargabi and A. N. Nusari, "Discovering vital patterns from UST students data by applying data mining techniques," *2010 2nd Int. Conf. Comput. Autom. Eng. ICCAE 2010*, vol. 2, no. 2, pp. 547–551, 2010.

[132] Deloitte, "Data analytics and workforce strategies New insights for performance improvement and tax efficiency," 2014.

[133] T. Issue, "Improve Military Personnel Planning With Analytics," 2011.

[134] B. L. Smith-proulx, "Job-Hunt LinkedIn for New Graduates."

[135] T. Ryan and S. Xenos, "Who uses Facebook? An investigation into the relationship between the Big Five, shyness, narcissism, loneliness, and Facebook usage," *Comput. Human Behav.*, vol. 27, no. 5, pp. 1658–1664, 2011.

[136] L. Qiu, H. Lin, J. Ramsay, and F. Yang, "You are what you tweet: Personality

expression and perception on Twitter," *J. Res. Pers.*, vol. 46, no. 6, pp. 710–718, 2012.

[137] S. Bai, B. Hao, A. Li, S. Yuan, R. Gao, and T. Zhu, "Predicting big five personality traits of microblog users," *Proc. - 2013 IEEE/WIC/ACM Int. Conf. Web Intell. WI 2013*, vol. 1, pp. 501–508, 2013.

[138] G. Chittaranjan, B. Jan, and D. Gatica-Perez, "Who's who with big-five: Analyzing and classifying personality traits with smartphones," *Proc. - Int. Symp. Wearable Comput. ISWC*, pp. 29–36, 2011.

[139] J. A. N. Cieciuch, "the Big Five and Belbin," 2014.

[140] A. C. E. S. Lima and L. N. De Castro, "Multi-label semi-supervised classification applied to personality prediction in tweets," *Proc. - 1st BRICS Ctries. Congr. Comput. Intell. BRICS-CCI 2013*, pp. 195–203, 2013.

[141] B. Minaei-Bidgoli, "Data Mining for a Web-Based Educational System," p. 267, 2004.

[142] R. Shaun, J. De Baker, and P. S. Inventado, "Chapter 4: Educational Data Mining and Learning Analytics," *Springer*, vol. Chapter 4, pp. 61–75, 2014.

[143] S. Trivedi, Z. Pardos, G. Sárközy, and N. Heffernan, "Spectral Clustering in Educational Data Mining.," *Proc. 4th Int. Conf. Educ. Data Min.*, pp. 129–138, 2011.

[144] A. Acharya and D. Sinha, "Application of Feature Selection Methods in Educational Data Mining," *Int. J. Comput. Appl.*, vol. 103, no. 2, pp. 34–38, 2014.

[145] N. Ahmed, D. A. City, U. A. Emirates, D. A. City, and U. A. Emirates, "Analyzing Undergraduate Students ' Performance in Various Perspectives using Data Mining

Approach," vol. 3, no. 8, pp. 59–65, 2013.

[146] C. M. Bishop, *Pattern Recognition and Machine Learning*, vol. 4, no. 4. 2006.

[147] U. Gupta and N. Chatterjee, "Personality Traits Identification Using Rough Sets Based Machine Learning," *2013 Int. Symp. Comput. Bus. Intell.*, pp. 182–185, 2013.

[148] and C.-J. L. Chih-Wei Hsu, Chih-Chung Chang, "A Practical Guide to Support Vector Classification," *BJU Int.*, vol. 101, no. 1, pp. 1396–400, 2008.

[149] C. C. J. C. Burges, "A Tutorial on Support Vector Machines for Pattern Recognition," *Data Min. Knowl. Discov.*, vol. 2, no. 2, pp. 121–167, 1998.

[150] N. T. N. Hien and P. Haddawy, "A decision support system for evaluating international student applications," *Proc. - Front. Educ. Conf. FIE*, pp. 1–6, 2007.

[151] R. Jindal and M. D. Borah, "A Survey on Educational Data Mining and Research Trends," *Int. J. Database Manag. Syst.*, vol. 5, no. 3, pp. 53–73, 2013.

[152] R. S. J. D. Baker, "Data mining for education," *Int. Encycl. Educ.*, vol. 7, pp. 112–118, 2010.

[153] S. Adali and J. Golbeck, "Predicting Personality with Social Behavior," *2012 IEEE/ACM Int. Conf. Adv. Soc. Networks Anal. Min.*, pp. 302–309, 2012.

[154] G. Mohammadi and A. Vinciarelli, "Automatic personality perception: Prediction of trait attribution based on prosodic features," *IEEE Trans. Affect. Comput.*, vol. 3, no. 3, pp. 273–284, 2012.

[155] O. Celiktutan, E. Sariyanidi, and H. Gunes, "Let me tell you about your personality!†: Real-time personality prediction from nonverbal behavioural cues," *2015 11th IEEE Int. Conf. Work. Autom. Face Gesture Recognition, FG 2015*, p. 6026, 2015.

[156] F. Celli and L. Polonio, “Relationships between personality and interactions in facebook,” *Soc. Netw. Recent Trends, Emerg. Issues Futur. Outlook*, pp. 41–53, 2013.

[157] D. Stillwell and M. Kosinski, “The personality of popular facebook users,” *Proc. ACM 2012 Conf. Comput. Support. Coop. Work*, pp. 955–964, 2012.

[158] G. Saucier and L. R. Goldberg, “The Language af Personality : Lexical Perspectives on the Five-Factor Model,” *Five-Factor Model Personal. Theor. Perspect.*, pp. 21–50, 1996.

[159] L. Goldberg, “From Ace to Zombie: Some explorations in the language of personality,” *Advances in personality assessment*, vol. 1. pp. 203–234, 1982.

[160] O. Celiktutan and H. Gunes, “Automatic Prediction of Impressions in Time and across Varying Context: Personality, Attractiveness and Likeability,” *IEEE Trans. Affect. Comput.*, vol. 3045, no. January 2016, pp. 1–1, 2016.

[161] D. P. Schmitt, J. Allik, R. R. McCrae, and V. Benet-Martinez, “The Geographic Distribution of Big Five Personality Traits: Patterns and Profiles of Human Self-Description Across 56 Nations,” *J. Cross. Cult. Psychol.*, vol. 38, no. 2, pp. 173–212, 2007.

[162] T. A. Byington, “Communities of Practice: Using Blogs to Increase Collaboration,” *Interv. Sch. Clin.*, vol. 46, no. 5, pp. 280–291, 2011.

[163] A. Abraham, *Computational social networks: Mining and visualization*, vol. 9781447140. 2012.

[164] R. M. Branch, “Online learning analytics on social networking sites: how to tap the potential of data mining in research of educational technology.”

[165] A. Jalal and Y. Zaidieh, “The Use of Social Networking in Education: Challenges and Opportunities,” *World Comput. Sci. Inf. Technol. J.*, vol. 2, no. 1, pp. 2221–741, 2012.

[166] A. Kaklauskas, *Biometric and Intelligent Decision Making Support*, vol. 81. 2015.

[167] D. Markovikj, S. Gievska, M. Kosinski, and D. Stillwell, “Mining Facebook data for predictive personality modeling,” *Proc. 7th Int. AAAI Conf. Weblogs Soc. Media (ICWSM 2013), Boston, MA, USA*, pp. 23–26, 2013.

[168] M. M. S. Education, “Final Report A Survey of K-12 Educators on Social Networking and Content-Sharing Tools,” *Education*, vol. 8, no. 1, pp. 32–61, 2009.

[169] “Educational Data Mining: Potentials for 20th Century Learning Science,” *Science (80-.).*

[170] E. Lee, J. Ahn, and Y. J. Kim, “Personality traits and self-presentation at Facebook,” *Pers. Individ. Dif.*, vol. 69, pp. 162–167, Oct. 2014.

[171] B. R. Lambiotte and M. Kosinski, “Tracking the Digital Footprints of Personality,” vol. 102, no. 12, pp. 1934–1939, 2014.

[172] D. Quercia, M. Kosinski, D. Stillwell, and J. Crowcroft, “Our Twitter Profiles, Our Selves: Predicting Personality with Twitter,” *2011 IEEE Third Int’l Conf. Privacy, Secur. Risk Trust 2011 IEEE Third Int’l Conf. Soc. Comput.*, pp. 180–185, Oct. 2011.

[173] T. L. Giluk and B. E. Postlethwaite, “Big Five personality and academic dishonesty : A meta-analytic review,” *Pers. Individ. Dif.*, vol. 72, pp. 59–67, 2015.

[174] P. Lison, *An introduction to machine learning*. 2015.

[175] S. Shalev-Shwartz and S. Ben-David, *Understanding Machine Learning*. 2014.

[176] E. Alpaydin, *I2Ml*. .

[177] E. Alpaydın, “Introduction to machine learning,” *Methods Mol. Biol.*, vol. 1107, pp. 105–128, 2014.

[178] M. I. Jordan and T. M. Mitchell, “Machine learning: Trends, perspectives, and prospects,” *Science (80-.).*, vol. 349, no. 6245, pp. 255–260, 2015.

[179] P. Haller and H. Miller, “Parallelizing Machine Learning–Functionally,” *Scala Work.*, 2011.

[180] a. Srivastava, E.-H. S. H. E.-H. S. Han, V. Singh, and V. Kumar, “Parallel formulations of decision-tree classification algorithms,” *Proceedings. 1998 Int. Conf. Parallel Process. (Cat. No.98EX205)*, vol. 24, pp. 1–24, 1998.

[181] X. Pan and C. Sciences, “Parallel Machine Learning Using Concurrency Control,” 2017.

[182] C. Lin, “Optimization and Machine Learning,” 2013.

[183] T. Hastie, R. Tibshirani, and J. Friedman, “The Elements of Statistical Learning,” *Math. Intell.*, vol. 27, no. 2, pp. 83–85, 2001.

[184] T. Hastie, “K-Fold Cross-Validation,” *Sldm 3*, pp. 3–11, 2009.

[185] M. F. Uddin, S. Rizvi, and A. Razaque, “Proposing Logical Table Constructs for Enhanced Machine Learning Process,” *IEEE Access*, vol. 6, pp. 1–1, 2018.

[186] M. A. Khan, M. F. Uddin, and N. Gupta, “Seven V’s of Big Data understanding Big Data to extract value,” *Proc. 2014 Zo. 1 Conf. Am. Soc. Eng. Educ.*, pp. 1–5, Apr. 2014.

[187] M. F. Uddin and J. Lee, “Utilizing Relevant Academic and Personality Features from Big Unstructured Data to Identify Good and Bad Fit Students,” *Procedia*

Comput. Sci., vol. 95, pp. 383–391, 2016.

[188] M. F. Uddin and J. Lee, "We Are What We Generate - Understanding Ourselves Through Our Data," *Procedia Comput. Sci.*, vol. 95, pp. 335–344, 2016.

[189] M. F. Uddin, "A framework to identify educational relevance in social networking posts," *2016 IEEE 7th Annu. Ubiquitous Comput. Electron. Mob. Commun. Conf. UEMCON 2016*, 2016.

[190] P. Ave, "Noise Removal and Structured Data Detection to Improve Search for Personality Features," no. 2011, pp. 1349–1355, 2016.

[191] M. F. Uddin, S. Banerjee, and J. Lee, "Recommender system framework for academic choices: Personality based recommendation engine (PBRE)," in *Proceedings - 2016 IEEE 17th International Conference on Information Reuse and Integration, IRI 2016*, 2016.

[192] M. F. Uddin and J. Lee, "Proposing stochastic probability-based math model and algorithms utilizing social networking and academic data for good fit students prediction," *Soc. Netw. Anal. Min.*, vol. 7, no. 1, p. 29, Jul. 2017.

[193] Z. Tang and J. Maclennan, "Data Mining With SQL Server 2005," p. 483, 2005.

[194] G. S. Linoff, *Data Analysis using SQL and Excel*. 2008.

[195] G. Fouché and L. Langit, "Data Mining with Excel," in *Foundations of SQL Server 2008 R2 Business Intelligence*, Berkeley, CA: Apress, 2011, pp. 301–328.

Appendix

We have utilized the data from the following sources listed below. Some datasets were raw files, CSV, Excel, and SQL lite format with parameters and field definitions. We transformed some of the input data into the SQL Server data warehouse. Some of datasets are found to be ideal for doing healthcare preventive medicine, stock market, epidemic, and crime control prediction.

Appendix A1. eMLEE Resources

Introduction, Current/Future Publications links, and Future work progress will be available on the blog address @ https://emlee-phd.blogspot.com/2018/03/enhanced-machine-learning-engine.html

Appendix A2. Dataset Sources

1. http://www.Kaggle.com
2. http://snap.standford.edu
3. https://docs.google.com/forms/d/1l57Un32YH6SkltntirUeLVpgfn33BfJuFLcYupg43oE/viewform?edit_requested=true, online questionnaire from students
4. http://archive.ics.uci.edu/ml/index.php
5. https://aws.amazon.com/datasets/
6. https://cloud.google.com/bigquery/public-data/. We are experimenting it using BigQuery in our Sandbox environment and will publish results in future.
7. https://www.reddit.com/r/bigquery/wiki/datasets
8. https://docs.microsoft.com/en-us/azure/sql-database/sql-database-public-data-sets.

Appendix A3. Tools

1. **For Data Hosting -** Microsoft SQL Server[193] (Business Intelligence, SQL Server Analysis Services, and Data mining) as our data warehouse.

2. **For MS ML -** Microsoft Azure machine learning tools.
3. **For Charting** – MatPlotLib, Microsoft Excel data mining tools[194]**,** [195], Graphpad prism, SigmaPlot, Matplotlib, Plotly.
4. **For Algorithm basic testing -** Microsoft C# (mostly for learning in the beginning)
5. **For Python Scientific, Math and ML Packages and libraries -** Pandas, NumPy, SciPy, scikit-learn
6. **For Development Studio** – Python 3.5, IPython, Python Jypyter Notebook, Anaconda, VS, R-Studio.

Appendix A4. Libraries

Python:

SL standard methods

1. from sklearn import tree
2. from sklearn.svm import SVC
3. sklearn.linear_model.LogisticRegression
4. from sklearn.ensemble import RandomForestClassifier
5. from sklearn.ensemble import ExtraTreesClassifier
6. from sklearn.tree import DecisionTreeClassifier
7. from sklearn import model_selection
8. from sklearn.linear_model import LinearRegression

SL boosting methods

1. from sklearn import ensemble
2. from sklearn.ensemble import GradientBoostingRegressor
3. from sklearn.ensemble import AdaBoostClassifier
4. sklearn.ensemble.GradientBoostingClassifier
5. sklearn.discriminant_analysis.LinearDiscriminantAnalysis
6. from sklearn.linear_model import ElasticNet
7. from sklearn.datasets import make_regression

Datasets

1. from sklearn.datasets import make_blobs
2. from sklearn.datasets import load_iris
3. from sklearn import datasets
4. from sklearn import datasets, linear_model
5. from sklearn import svm, datasets
6. from sklearn.model_selection import train_test_split

Validation

1. from sklearn.model_selection import cross_val_score
2. FSelector-package,
3. sklearn.feature_extraction,
4. sklearn.decomposition

Performance metrics

1. from sklearn.metrics import precision_score, \
2. recall_score, confusion_matrix, classification_report, \
3. accuracy_score, f1_score
4. from sklearn.metrics import matthews_corrcoef,
5. sklearn.metrics.auc
6. metrics.roc_curve
7. from sklearn.decomposition import PCA
8. from scipy.stats import chisquare
9. sklearn.metrics.cohen_kappa_score
10. sklearn.linear_model.Lasso
11. from sklearn.feature_selection import RFE
12. from sklearn.svm import SVR
13. from sklearn.utils import shuffle
14. from sklearn.metrics import mean_squared_error
15. from sklearn.metrics import precision_recall_curve
16. sklearn.metrics.accuracy_score

R:

1. randomForest
2. caret
3. RWeka
4. mBoost
5. tree