International Journal of Engineering & Technology, 7 (1.1) (2018) 143-149

International Journal of Engineering & Technology

Website: www.sciencepubco.com/index.php/IJET

Research paper

Machine learning algorithms: a background artifact

J. Deepika [1]*, T. Senthil [2], C. Rajan [3], A. Surendar [4]

[1]*Assistant Professor, Department of Information Technology, K.S.Rangasamy College of Technology*
[2]*Assistant Professor, Department of ECE, K S R Institute for Engineering and Technology*
[3]*Associate Professor, Department of Information Technology, K.S.Rangasamy College of Technology*
[4]*Assistant Professor, School of Electronics, Vignan's Foundation for Science, Research & Technology*
**Corresponding author E-mail: deepi.realmail@gmail.com*

Abstract

With the greater development of technology and automation human history is predominantly updated. The technology movement shifted from large mainframes to PCs to cloud when computing the available data for a larger period. This has happened only due to the advent of many tools and practices, that elevated the next generation in computing. A large number of techniques has been developed so far to automate such computing. Research dragged towards training the computers to behave similar to human intelligence. Here the diversity of machine learning came into play for knowledge discovery. Machine Learning (ML) is applied in many areas such as medical, marketing, telecommunications, and stock, health care and so on. This paper presents reviews about machine learning algorithm foundations, its types and flavors together with R code and Python scripts possibly for each machine learning techniques.

Keywords: *Machine learning algorithms and types, supervised learning algorithms, unsupervised learning algorithms, R code, python script.*

1. Introduction

ML denotes to the methods tangled in distributing through massive facts in the greatest intellectual way to arise better understandings. ML algorithms are defined to be culturing an objective function (f) which better draws input identifier (g) to an output identifier (h) as in equation 1[1].

$$h = f(g) \quad (1)$$

This future output prediction is not that much easier to do manually. Hence an automated system is expected to do the process [1]. Thus use of machine learning algorithms come into the scene. For every new input (g) the output (h) is predicted genuinely using machine learning algorithms. This state is said to be predictive molding/predictive analytics. The major operation is to assess he most possible predictions with the present data. Each data is segregated as training set and testing set as in figure 1.

Five basic steps for a ML task

1. Data accumulation: Data gathered from various sources are used for analysis.
2. Data preprocessing: Before getting into the actual processing of data, preprocessing is mandatory. This step is used to noise or other unwanted data from the gathered data.
3. Prototype training: This step contains selecting the suitable algorithm and depiction of data in a pattern (model) format. The preprocessed data is often divided into two parts namely training and testing data.
4. Pattern evaluation: In this step, the resultant pattern is validated for its correctness.
5. Performance enrichment: This step involves picking another different pattern with better efficiency. However substantial time is required in data gathering and training.

Despite any model/pattern, the above said five steps are mandatory to configure the ML technique.

1.2. Types of machine learning algorithms–scenario based

1.2.1. Supervised learning (SL) or prognostic models

This is used to assess the upcoming result with the help of chronological data [2]. These models are instructive as much concentration is emphasized in training phase [5]. For instance, SL is applied if a selling firm wishes to find its customers list. It could also be used in prediction of earthquakes, cyclones etc. to determine the Insurance credit. Few examples of these prediction algorithms are: Nearest neighbor, NaiveBayes, DecisionTrees (J48, Random Forest), Regression etc.

1.2.2. Unsupervised learning (UL) or evocative models

UL is suitable to train vivid models with no target and no sole feature is significant compared to one another [3]. For instance, UL is applied in case if a vender desires to find which product does the customer buys frequently. Moreover, in medicinal business, UL may be applied to envisage the diseases that may prone to occur laterally with diabetes. Few examples of UL based algorithms are Apriori algorithm, Simple K- means clustering

1.2.3. Reinforcement learning (RL)

RL is applied when the system is trained to yield decisions automatically with the business requirements only with a sole motto to exploit better effectiveness (performance) [2]. The underlying idea is a software agent is trained in an environment for problem solving. This repeated learning procedure promises lower human proficiency thus saving human effort [6]. One best example of RL algorithm is Markov Decision Process. RL fun-

damentally includes learning by relating using the situation. Thus it differs from SL in its characteristic.

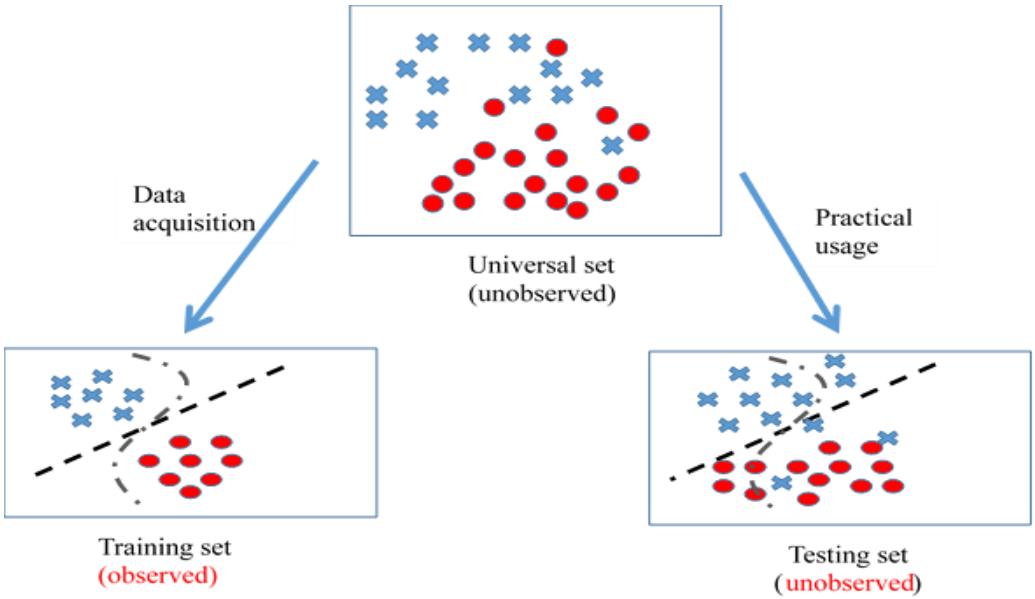

Fig. 1: Training and testing model in machine learning system

1.1Parametric and nonparametric algorithms

2.1. Parametric algorithms

In parametric algorithms, the function can be simplified to any recognized form called as parametric ML systems [3]. Generally, parametric procedures are divided into 2 steps:

1. Choosing an appropriate procedure of the function
2. Acquire the quantities of the function

Few examples of parametric ML algorithms are Linear Regression and Logistic Regression [13].

2.2. Non-parametric algorithms

On the other hand, processes with negative conventions for relating the parameters are called as nonparametric ML algorithms [3]. Without creating expectations, they are made free for training process. These are frequently agiler, attain restored accuracy. Though, the time for training the data is substantially very high. Some instances of nonparametric algorithms include Support Vector Machines (SVM) and Neural Networks (NN) [25].

2. Notion, divergence and tradeoffs

In general, ML algorithms are evident from notion, divergence and tradeoffs. Notion are shortening prospects for easy learning of the target function. In general, parametric algorithms are easy to learn and understand compared to nonparametric algorithms [2]. They possess less analytical routine on composite problems. Decision trees are considered to be less bias algorithm example and linear regression are considered to be high bias algorithm example.

Divergence is the quantity estimate where the target function drive to alteration with diverse training data. With training data, the algorithm must possess some divergence, without nil variance. The aim of any prognostic modelling ML algorithm is achieving low bias and low divergence. Likewise, the devised algorithm is expected to perform better. Figure 2 lists the various categories of ML algorithms. Each of these are discussed in the preceding sections.

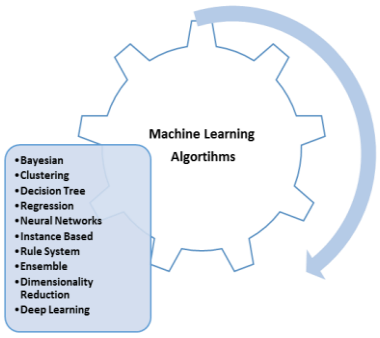

Fig. 2: Categories of machine learning algorithms

3. Linear algorithms

4.1. Linear regression

Linear regression is one well tacit algorithms in the area of statistics and machine learning. It is being used over 200 years. Predictive analysis is chiefly apprehensive when the error is minimized or when the prediction is accurate, at the expense of explainability. The linear regression is signified using an equality describing a line fitting the association amid the input (g) and the output (h), using coefficients weighting as in equation 2.

For example:

$$h = X0 + X1 * g \quad (2)$$

The output (h) can be predicted based on given input (g) and the major task is assigning perfect coefficients X0 and X1.

A better practice while using linear regression is removing correlated variables and noise from the given data, if possible. The K-Nearest Neighbors(KNN) algorithm stands as a finest instance of high-variance algorithm, and Linear Discriminant Analysis(LDA) stands as a finest instance of low-variance algorithm [24]. The parameterization of ML algorithms must handle notion and divergence. Notion and divergence are opposite poles where increasing the notion will decrease the divergence and vice versa.

Python script for implementing linear model

```
#Import necessary libraries like pandas, numpy
from sklearn import linearr_model
xx_train=input_variables
yy_train=target_variables
xx_test=input_variables
# Generate regression object
linearr = linearr_model.LinearRegression()
# Train the model and assess score
linearr.fit(xx_train, yy_train)
linearr.score(xx_train, yy_train)
print ('Coefficient:', linearr.coef_)
print ('Intercept: \n', linearr.intercept_)
# Output Prediction
predicted_output= linearr.predict(xx_test)
R Code for implementing linear model
#Load datasets
xx_train <- input_variables
yy_train <- target_variables
xx_test <- input_variables
h <- cbind (xx_train, yy_train)
# Train the model and assess score
linearr <- lm (yy_train ~ ., data = h)
summary (linearr)
#Output Prediction
predicted_output= predict (linearr,xx_test)
```

4.2. Logistic regression algorithm

Logistic regression is yet another ML technique in statistics. It is used for dual cataloging problems (difficulties with 2 class values). In this method, output prediction is distorted using a logistic function which is capable of transforming any value into the range 0 to 1. This probabilistic conversion simplifies to break ideals to 0 and 1 and guess a class value. This is more useful for those problems requiring greater justifications for prediction. Similar to linear regression, logistic regression fails to work in case of unrelated attributes and duplicate attributes.
Python script for implementing LogisticRegression

```
#Import necessary libraries
from sklearn.linearr_model import LogisticRegression
# Make object
Model1 = LogisticRegression()
# Train the model and assess score
Model1.fit (W,z )
Model1.score(W, z)
print ('Coefficient: ', Model1.coef_)
print ('Intercept: ', Model1.intercept_)
# Output Prediction
predicted_output= Model1.predict(xx_test)
R Code for implementing LogisticRegression
h <- cbind (xx_train,yy_train)
# Train the model and assess score
logistic_reg <- glm(yy_train ~ ., data = xx, status='binomial')
summary(logistic_reg)
# Output Prediction
predicted_output= predict (logistic_reg, xx_test)
```

4.3. Linear discriminant analysis(LDA) algorithm

As stated in the above section, Logistic Regression works better only for 2-class classification problems. In cases where classification algorithm with additional classes if present, then LDA algorithm is best suited (a brute-force approach). It contains algebraic assets of data, projected for each class. Every distinct input variable includes the following:

1. The Mean.
2. The Variance.

Estimates are completed by scheming a different cost of every class and predictions are made based on the thus attained biggest value.

4. Non-linear algorithms

5.1. Classification and regression trees algorithm

Decision Trees are better used for predictive modelling machine learning that are represented as a binary tree from algorithms and data structures. Each node signifies a particular input (x) and a splinter point. The leaves possess an output (y) for making prediction. Predictions are done based on dragging all beside the tree nodes via fragments till arriving the leaf. The leaf nodes output the class value. Generally, decision trees are believed to learn fast and even much very faster for making predictions. With much lesser pre-processing of data decision trees offer accurate for a broad range of problems. Decision trees possess high-divergence and be able to return further precise predictions in a group [10]. The technique adopts a Gaussian distribution (bell curve). Hence, it works better if outliers are removed from the data beforehand.
Python script for implementing Decision Trees

```
#Import necessary libraries like pandas, numpy
from sklearn import tree
# Generate object
Mode1l = tree. DecisionTreeClassifier (criterion='Infogain')
# Train the model and assess score
Model1.fit (W, z)
Model1.score(W, z)
#Output Prediction
predicted_output= Model1.predict(xx_test)
R code for implementing DecisionTrees [4]
library(rpart)
h <- cbind(xx_train,yy_train)
# cultivate tree
fitt <- rpart(yy_train ~ ., data = xx,method ="class")
summary(fitt)
#Output Prediction
predicted_output= predict (fitt,xx_test)
```

5.2. Naive bayes classification

NaiveBayes is a modest and powerful procedure suited in prognostic modeling. Two kinds of probabilities are encompassed from the calculations obtained through training data. They are as follows:

1. The probability value of every separate class.
2. The conditional probability value of every class specified for all x value.

After the individual and conditional probabilities are calculated, Naïve Bayes is adopted for predictions using Bayes Theorem for each independent input x values. Nevertheless, for real-valued data Gaussian distribution is used for approximating these probabilities [13]. Once if probabilities are calculated correctly, the method is identically efficient for huge complex problems.
Python script for Gaussian NaiveBayes algorithm

```
#Import necessary libraries
from sklearn naiveBayes import GaussianNB
# Generate SVM object model
# Train the model and assess score
Model1.fit (W, z)
#Output Prediction
predicted_output= Model1.predict(xx_test)
R code for Gaussian NaiveBayes algorithm [4]
library(e1071)
h <- cbind (xx_train, yy_train)
Bfit <-naiveBayes(yy_train ~ ., data = h)
summary(Bfit)
#Output prediction
predicted_output= predict (Bfit, xx_test)
```

5.3. KNearest neighbors(KNN) algorithm

KNN algorithm uses the whole training dataset for using K utmost alike occurrences (the neighbors). The method uses the Euclidean distance metric to estimate the likeliness. Yet, KNN entails much larger memory to accommodate all data. Classifying the data based on the closeness may lead to very high dimensionality breakdown. This is stated as curse of dimensionality [24]. So, output prediction must be done only based on the relevant input variable.
Python script for KNN algorithm

```
#Import necessary libraries
from sklearn neighbors import KNNClassifiers
# Generate KNN object model
KNNClassifiers (nn_neighbors = 7)
# Train the model and assess score
Model1.fitt (W, z)
#Output Prediction
predicted_output= Model1.predict(xx_test)
```

5.4. Learning vector quantization (LVQ)

The great impact of KNN is that always the modelling function depends on the entire data. LVQ is a non-natural neural network algorithm [13]. LVQ is represented as a group of codebook vectors. At the primary stage itself, the vectors are designated arbitrarily and improved to review the best output for a number of repetitions. The best appropriate neighbor is recognized using distance computation among individual codebook vector and the novel data case. The class cost for the unsurpassed matching part is later resumed as prediction. Preeminent outcomes are attained by rescaling the data between 0 and 1. KNN yields best results with dataset by accepting LVQ to diminish the memory of storing larger dataset.
R code for implementing LVQ

```
#LVQ
set.seed(7)
library(caret)
data(iris)
control <- trainControl(method="repeatedcv", number=10,
repeats=3)
grid <- expand.grid(size=c(5,10,15,20,25,30,35,40,45,50),
k=c(3,5))
model <- train  (Species~., data=iris, method="lvq",
trControl=control, tuneGrid=grid)
print(Model1)
plot(Model1)
```

5.5. SVM algorithm

SVM are possibly one among the prevalent ML algorithms. SVM possess a hyperplane that best distinct the points in the provided input variable space using class (class 0 or class 1) [8]. The diffidence among the hyperplane and the neighboring data points is the margin. Basically, an optimization algorithm finds the coefficient values [9]. It bids preeminent classification performance and offers best efficiency for perfect classification. Still, the method does not promise any stout guess on data and apt the data onto the input space [21] [22]. Data points cannot be classified in case where more than one SVM class accepts or rejects the data points.
Python script for implementing SVM

```
#Import necessary Libraries
from sklearn import svm
# Generate SVM object
Model1 = svm.svc()
# Train the model and assess score
Model1.fitt(W, z)
Model1.score(W, z)
#Output Prediction
predicted_output = Model1.predict(xx_test)
R code for implementing SVM [17]
library(e1071)
h <- cbind(xx_train, yy_train)
fitt <-svm(yy_train ~ ., data = h)
summary(fitt)
# Output Prediction
predicted_output = predict (fitt, xx_test)
```

5. Ensemble algorithms

6.1. Bagging and random forest

Random Forest is yet another popular and most powerful ensemble machine learning algorithms. Bootstrap is an influential statistical technique for approximating a magnitude on data. Bagging uses a similar approach but inherits numerous examples of training data from which the models are constructed. The predictions are made on the average of every individual model [13]. The models thus shaped are dissimilar yet much accurate.
Each tree is lodged & grown as follows:

- In N amount of situations, few sample are picked at arbitrary however with replacement which are considered to be training set for growing the tree.
- With M inputs, a distinct number m<<M is acknowledged for the best split to be done. This m value must persistent for the forest growing.
- Every tree is grownup for its leading promising height with no pruning.

R code for implementation of RandomForest algorithm

```
#Random Forest
# Load the party package.
# It will automatically load other required packages.
library(party)
library(randomForest)
library(forestFloor)
library(AUC)
# Create the forest.
output.forest <- randomForest(nativeSpeaker ~ age + shoeSize
+ score,  + data = readingSkills, keep.inbag = T,keep.forest = T)
# View the forest results.
print(output.forest)
Call:
randomForest(formula = nativeSpeaker ~ age + shoeSize +
score, data = readingSkills, keep.inbag = T, keep.forest = T)
print(importance(output.forest,type = 2))
 getTree(output.forest,3,labelVar = TRUE)
 ff = forestFloor(output.forest,readingSkills,binary_reg =
T,calc_np=T)
Col = fcol(ff,cols=2,outlier.lim = 2.5)
#2D Visualisation
plot(ff,col=Col,plot_GOF = T)
#3D Visualisation
show3d(ff,1,col=Col,plot_GOF = T)
library(rgl)
rgl.snapshot("picture.png")
```

6.2. Boosting and adaboost

Boosting is an ensemble technique that generates a robust classifier out of feeble classifiers. A model is constructed from the training data, then engendering a next model by correcting the errors in previous model [14].
AdaBoost is a successful boosting algorithm recognized for binary classification that are used with smaller decision trees [7] [16]. The tree's performance for each training example is used to shape the subsequent tree without error [15]. The models are produced consecutively with updated weights on the training instances. The process is continued for many iterations until the last static model is obtained. Once all the trees are completed, the individual tree's performance is weighted on the training data

[12]. Outliers are to be removed as error correction is emphasized at each renewal.

R Code for implementing Boosting algorithm

R supports many boosting algorithms. However, here one such execution that uses decision trees as base classifiers are used. Therefore, the rpart package much be weighed down. Together the ada package could be used. With X be the matrix of features and class labels 0-1 then the command is

boostModel1 <- ada (x=X, y=class_labels)

6. Dimensionality reduction (DR) algorithms

Data are being captured in greater extent in day to day life. For these situations, DR algorithms could be used along with various other algorithms like Decision Tree, RandomForest, PCA, Factor Analysis for prediction.

Scenario: The Wine Data Set from the UCI Machine Learning Repository is considered. This data set contains the results of chemical analysis of 178 different wines from three cultivars. There observations contain the quantities of 13 constituents found in each of the three types of wines. The wine dataset is included in the HDClassif package, so let's install that and examine the dataset.

PCA using R

```
install.packages("HDclassif")
library(HDclassif)
data(wine)
str(wine)
install.packages("stats")
library(stats)
wine_pca <- prcomp(wine, center = TRUE, scale = TRUE)
summary(wine_pca)
#Visualizing the data set
biplot(wine_pca)
```

7. Neural networks (NNs)

NNs are well suited with big data analysis. However, the computational cost is very high for handling such data [19] [20]. NNs are just the interconnected nodes that corresponds to the input signals.

R Code Example for Neural Networks

```
library(iris)
field(infer)
library(neuralnetwork)
nn <- neuralnetwork (case~name+parity+induced+spontaneous,
data=infer, hidden=2, error.fct="ment", linear.output=FALSE)
nn field(nn)
# resultant matrix
nn$resultant.Matrix
output <- cbind ( nn$covariate, nn$net.resultant [[1]])
 dimensionnames(output) <- list (NULL, c ("age", "pari
ty","induced","spontaneous","nn-output"))
 head(output)
 head(nn$generalized.weights[[1]])
 plot(nn)
```

8. Rule systems

Rule-based systems or Rule system uses a distinct way to access data that are used artificial intelligence applications, domain-specific expert system. For instance, an expert system aids doctors for easy diagnosis of diseases. They find their applications towards natural language processing [23]. Rule are extracted automatically from the data to take decisions using many indirect methods. Few rule based algorithms are Cubist, OneRule, ZeroRule, etc.

R code for Cubist algorithm

```
library(cubist_rule)
library(mllbench)
data(iris)
iris$chas <- as.numeric(iris$chas) - 1
set.seed(1)
Train <- sample (1: nrow(iris), floor(.8*nrow(iris)))
Train_Predictors <- iris[Train, -10]
Test_Predictors <- iris[Train, -10]
Result <- iris$medv[Train]
Test_Outcome <- iris$medv[Train]
Tree1 <- cubist_rule (xx = Train_Predictors,
yy = Test_Outcome)
Tree1 summary(Tree1)
```

9. Deep learning algorithms

Deep learning is a machine learning technique that tries to model high-level abstractions in data using multiple processing layers [18]. For instance, with an image data the output is represented as vector values per pixel, edges or, regions of particular shape, etc. Deep learning works better with competent algorithms in classification or clustering. Numerous deep learning designs such as deep- neural networks, Convolutional Neural Networks(CNN), deep-belief networks and recurrent-neural networks applied to many domains like computer-vision, speech-recognition, NLP, audio-recognition and bio-informatics. Few deep learning algorithms are Bolzman Machine, Deep Belief Networks, CNN, Stacked Auto Encoders, etc.

R code for CNN to recognize handwritten images

```
train_data <- read.csv ( "imageset.csv")
mm = matrix(unlist(train_data [10,-1]), nrow = 28, byrow =
TRUE)
 image(mm,color=grey.colors(0-255))
 rotate <- function(x) t(apply(x, 2, rev))
paramater(row=c(2,3))
lapply(1:6,
function(x) image(rotate(matrix(unlist(train_data [x,-1]),nrow
= 28, byrow = TRUE)), color=grey.colors(0,255),
xlab=train_data[x,1]      ))
parameter (row=c(1,1))
library (caret)
Train<- Split (train_data$label, k=0.8, list=FALSE)
Train<-train_data .data[Train]
Test<-train_data .data [Train]
write.csv (train, file = "image1.csv", row.names = FALSE)
write.csv (test, file = "image2.csv", row.names = FALSE)
library(h2o)
local.h2o <- h2o.initilaize(ip = "localhost:8080", port = 23444,
startH2O = TRUE, nthreads=-1)
train<- read.csv ("image1.csv")
test  <- read.csv ("image2.csv")
  train_data[,1]<-as.factor(train_data [,1])
ttrData<-as.h2o(train)
ttsData<-as.h2o(test)
result.dl <- h2o.deeplearning(h = 2:785, yy = 1, ttrData, active
tion = " ", hidden=repres(150,5),epochs = 30)
predict.dl<-h2o.predict(object=result.dl, newdata=ttsData[,-1])
predict.dl.df<-as.data.frame(predict.dl)
summary(predict.dl)
testnew_labels<-test[,1]
 summary(diagram(table(test_labels, predict.dl.df[,1])))
test<-read.csv("image2.csv")
 test_h2o<-as.h2o(test)
 ddf.test <- as.data.frame(predict.dl.test)
ddf.test <- data.frame(ImageId =
seq(1,length(df.test$predict)),
Label = df.test$predict)
write.csv(ddf.test, file = "summary.csv", row.names=0)
 h2o.shutdown (prompt = 0)
```

10. Conclusion

ML is an extremely influential tool for solving many of the real-world problems. The primary aim of ML researchers is to propose a better and smart learning systems that reduces human effort. However, ML is emphasized towards data handling and analysis rather than time/cost complexity. These automated ML algorithms obviously reduces human error and effort. As ML algorithms are entirely big data-oriented they are continually grander and trustful thus used for direct programming. The sole overhead involved is to have structured data so that every individual ML algorithm might handle itself.

References

[1] Chang A, "R for Machine Learning, Prediction: Machine Learning and Statistics", *MIT OpenCourseWare*, (2012), pp.1-8.

[2] Singh A, Thakur N & Sharma A, "A review of supervised machine learning algorithms", *3rd International Conference on Computing for Sustainable Global Development (INDIACom)*, (2016), pp.1310-1315.

[3] Archana S & Elangovan K, "Survey of Classification Techniques in Data Mining", *International Journal of Computer Science and Mobile Applications*, Vol.2, (2014), pp.65-71.

[4] Sun B, Chen S, Wang J & Chen H, "A robust multi-class AdaBoost algorithm for mislabelled noisy data", *Knowledge-Based Systems*, Vol.102, (2016), pp.87-102.

[5] Chapelle O, Sindhwani V & Keerthi SS, "Optimization Techniques for Semi-Supervised Support Vector Machines", *Journal of Machine Learning Research*, Vol.9, (2013), pp.203–233.

[6] Chen D, Tian Y & Liu X, "Structural nonparallel support vector machine for pattern recognition", *Pattern Recognition*, Vol.60, (2016), pp.296-305.

[7] Blanco F, Ávila JC, Jiménez GR, Carvalho A, Díaz AO & Bueno RM, "Online adaptive decision trees based on concentration inequalities", *Knowledge-Based Systems*, Vol.104, (2016), pp.179-194.

[8] Fatima M & Pasha M, "Survey of Machine Learning Algorithms for Disease Diagnostic", *Journal of Intelligent Learning Systems and Applications*, Vol.9, (2017), pp.1-16.

[9] Xu J, Wu Q, Zhang J & Tang Z, "Exploiting Universum data in AdaBoost using gradient descent", *Image and Vision Computing*, Vol.32, (2014), pp.550-557.

[10] Das K & Behera RN, "A Survey on Machine Learning: Concept, Algorithms and Applications", *International Journal of Innovative Research in Computer and Communication Engineering*, Vol.5, No.2, (2017), pp.1301-1309.

[11] Miller LD & Soh LK, "Cluster-Based Boosting", *IEEE Transactions on Knowledge and Data Engineering*, Vol.27, (2015), pp.1491-1504.

[12] Baig M, Awais MM & El-Alfy EM, "AdaBoost-based artificial neural network learning", *Neurocomputing*, Vol.16, (2017), pp.22–41.

[13] Praveena M & Jaiganesh V, "A Literature Review on Supervised Machine Learning Algorithms and Boosting Process", *International Journal of Computer Applications*, Vol.169, No.8, (2017), pp.32-35.

[14] Pedregosa F, Varoquaux G, Gramfort A, Michel V, Thirion B, Grisel O, Blondel M, Prettenhofer P, Weiss R, Dubourg V & Vanderplas J, "Scikit-learn: Machine learning in Python", *Journal of Machine Learning Research*, (2011), pp.2825-2830.

[15] Xiao Q, Liang Y, Lu L, Yan S & Tai YW, "Exploring Heterogeneous Algorithms for Accelerating Deep Convolutional Neural Networks on FPGAs", *Proceedings of the 54th Annual Design Automation Conference*, (2017).

[16] Hiregoudar SB, Manjunath K & Patil KS, "A Survey: Research Summary on Neural Networks", *International Journal of Research in Engineering and Technology*, Vol.03, Special Issue.03, (2014), pp.385-389.

[17] Sharma V, Rai S & Dev A, "A Comprehensive Study of Artificial Neural Networks", *International Journal of Advanced Research in Computer Science and Software Engineering*, Vol.2, No.10, (2012), 278 -284.

[18] Utkin V & Zhuk YA, "An one-class classification support vector machine model by interval-valued training data", *Knowledge-Based Systems*, Vol.120, (2017), pp.43-56.

[19] Vijayarani S & Dhayanand S, "Liver Disease Prediction using SVM and Naïve Bayes Algorithms", *International Journal of Science, Engineering and Technology Research*, Vol.4, (2015), pp.816-820.

[20] Wang J, Jebara T & Chang SF, "Semi-supervised learning using greedy max- cut", *Journal of Machine Learning Research*, Vol.14, No.1, (2013), pp.771-800.

[21] Peng X, Rafferty K & Ferguson S, "Building support vector machines in the context of regularized least squares", Neuro computing, Vol.211, (2016), pp.129-142.

[22] Surendar, A., Arun, M., Basha, A.M. "Micro sequence identification of bioinformatics data using pattern mining techniques in FPGA hardware implementation" (2016), Asian Journal of Information Technology, 15 (1), pp. 76-81.

[23] Panchumarthi, G.P., Surendar,A."A review article on Fin-FET based self-checking full adders",(2017) Journal of Advanced Research in Dynamical and Control Systems, 9 (4), 8 p.

[24] Priyanka Reddy, G.S., Surendar, A."A review article on performance comparison of CNTFET based full adders",(2017) Journal of Advanced Research in Dynamical and Control Systems, 9 (4), pp. 9-20.

[25] Selvi, N., Surendar, A. "Efficient power reduction and glitch free mux based digitally controlled delay line",(2015), International Journal of Applied Engineering Research, 10 (10), pp. 9655-9659.

[26] Vishnu, S., Vignesh, S., Surendar, A."Design and implementation of ZETA micro-inverter for solar PV application"(2017), International Journal of Mechanical and Production Engineering Research and Development, 7 (4), pp. 215-222.

[27] Lakshmi, K., Surendar, A. "Verification of axiprotocol using system Verilog", (2017), International Journal of Mechanical Engineering and Technology, 8 (5), pp. 588-595.

[28] Surendar, A., Kavitha, M. "Secure patient data transmission in sensor networks", (2017), Journal of Pharmaceutical Sciences and Research, 9 (2), pp. 230-232.

[29] Surendar, A."FPGA based parallel computation techniques for bioinformatics applications",(2017) International Journal of Research in Pharmaceutical Sciences, 8 (2), pp. 124-128.

[30] Surendar, A."Evolution of gait biometric system and algorithms-A review" (2017) Biomedical and Pharmacology Journal, 10 (1), pp. 467-472.

[31] Vimalkumar, M.N., Helenprabha, K., Surendar, A."Classification of mammographic image abnormalities based on emo and LS-SVM techniques",(2017) Research Journal of Biotechnology, 12 (1), pp. 35-40.

[32] Manju, K., Sabeenian, R.S., Surendar, A."A review on optic disc and cup segmentation",(2017) Biomedical and Pharmacology Journal, 10 (1), pp. 373-379.

[33] Surendar, A., Rani, N.U."High speed data searching algorithms for DNA searching",(2016) International Journal of Pharma and Bio Sciences, 2016 (Special Issue), pp. 73-77.

[34] Surendar, A., Arun, M."Efficient DNA sequence analysis for reduced gene selection using frequency analysis", (2016) Journal of Chemical and Pharmaceutical Sciences, 9 (4), pp. 3367-3373.

[35] Surendar, A., George, A."A real-time searching and sequencing assembly platform based on an FPGA implementation for Bioinformatics applications",(2016) International Journal of Pharma and Bio Sciences, 7 (4), pp. B642-B647.

[36] Surendar, A., Arun, M."FPGA based multi-level architecture for next generation DNA sequencing",(2016) Biomedical Research (India), 2016, pp. S75-S79.

[37] Surendar, A., Arun, M., Basha, A.M."Micro sequence identification of bioinformatics data using pattern mining techniques in FPGA hardware implementation",(2016) Asian Journal of Information Technology, 15 (1), pp. 76-81.

[38] Prabu, G., Surendar, A."Virus detection by using a pattern matching algorithm for network security",(2015) International Journal of Applied Engineering Research, 10 (10), pp. 9565-9569.

[39] Surendar, A., Arun, M., Periasamy, P.S."Hardware based algorithms for bioinformatics applications - A survey",(2013) International Journal of Applied Engineering Research, 8 (6), pp. 745-754.

[40] B. Saichandana, G. Rachana sri, A. Surendar and B. Suniltej "controlling of wall lamp using arduino", International Journal of Pure and Applied Mathematics, Volume 116 No. 24 2017, 349-354, ISSN: 1311-8080 (printed version); ISSN: 1314-3395 (online version)

[41] A. Surendar and Usha Rani Nelakuditi, "Editorial - New Developments in Electronics, Cloud and IoT", Electronic Government, An International Journal, Vol. 13, No. 4, 2017, ISSN online: 1740-7508 ISSN print: 1740-7494,pp -287-289

[42] Surendar, A. "Improving Age Invariant Face Recognition System Using Facial Features." Research Journal of Pharmacy and Technology10.6 (2017): 1762-1766.
[43] Sahu, Anil Kumar, Rashid Sheikh, and A. Surendar. "Ultra low power design approach of asynchronous delta sigma modulator." International Journal of Engineering & Technology 8.1.1 (2018): 84-87.
[44] Surendar, A., M. Kavitha, and V. Saravanakumar. "Proactive model based testing and evaluation for component-based systems." International Journal of Engineering & Technology 8.1.1 (2018): 74-77.