Machine Learning Based Text Document Classification for E-Learning

Benny Thomas, Chandra J

***Abstract.**: The number of e-learning websites as well as e-contents are increasing exponentially over the years and most of the time it become cumbersome for a learner to find e-content suitable for learning as the learner gets overwhelmed by the enormity of the content availability. The proposed work focus on evaluating the efficiencies of the different classification algorithm for the identification of the e-learning content based on difficulty levels. The data is collected from many e-learning web sites through web scraping. The web scraper downloads the web pages and parse to text file. The text files were made to run through many machine learning classification algorithms to find out the best classification model suitable for achieving the highest score with minimum training and testing time. This method helps to understand the performance of different text classification algorithms on e-learning contents and identifies the classifier with high accuracy for document classification.*

***Keywords:** E-Learning, Text Document, Difficulty levels. Classifications.*

I. INTRODUCTION

As the number of online learners increased, the online content availability also increased dramatically. Finding the suitable learning material become too difficult from the large collection of content and the contents are so skewed that the learner find it too difficult to get the suitable learning material. Different methods were used to address this issue. One prominent method is recommended systems which helps to find items from a large collection of items. Recommended systems are used in different domains such as entertainments, e-learning, news, e-commerce etc. G. Geetha et al., mention about different types of systems such as hybrid filtering, content based filtering and collaborative filtering used in Recommend systems[1]. K. I. Ghauth and N. A. Abdullah, uses one type of content based filtering based on already studied learners' rating is proposed that can bring additional similar learning material through content based filtering and learners resource help to improve the comprehension Already studied learner's resource is equal to collaborative filtering approach. Therefore these approach uses content, collaborative and hybrid filtering methods for recommendation. [2] . However the recommended system does not classify them based on the cognitive levels and classification of e-learning documents based on difficulty level is not addressed.

Revised Manuscript Received on January 15, 2020

***** Correspondence Author

Benny Thomas*, Department of Computer Science, Christ University, Bangalore, India.

Chandra J, Department of Computer Science, Christ University, Bangalore, India.

In the current work intelligent cognitive level identifier framework is proposed which classifies the e-learning content based on its difficulty levels. The websites urls are used as inputs to get the e-content from the web site. The framework collects input data through web scraping. The web scraper downloads the web pages and parse to text file. The text files were made to run through many machine learning classification algorithms to find out the best classification model suitable for achieving the highest score with minimum training and testing time.

The remaining part of the paper is organized as follows, Section II. Literature review of the existing work in e-learning classification, Section III.Methodology of the proposed work with a study of different classification algorithms, Section IV. Implementation of the model V. Result and Discussions and Section VI. Conclusion.

II. LITERATURE REVIEW

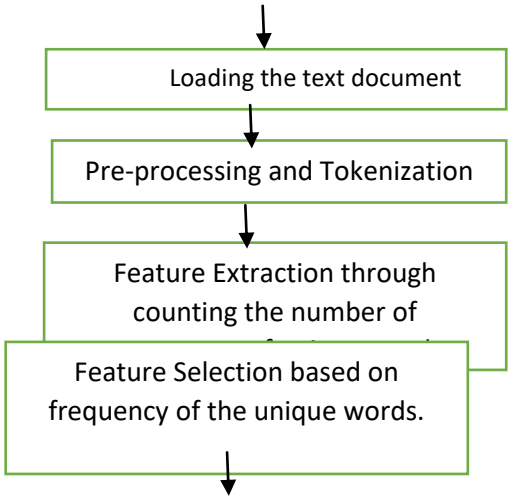
Numerous studies have conducted to attain suitable text classifications and recommendation in e-learning space. Sankar Perumal et al., proposes a new method for recommendation system which provides suitable contents by refining the final frequent item patterns evolving from frequent pattern mining technique and then classifying the final contents using fuzzy logic into three levels [3]. K. Tarus and Zhendong Niu conducted a study to demonstrate that ontology use for representing knowledge in e-learning recommender systems can bring improvement in the quality of recommendations. The Hybridization of recommended system can also improve the quality of recommendation[4]. P. G. Desai et al., proposed one of the most efficient method in text classification is using baysian classifiers which uses statistical and supervised learning method using machine learning algorithms. The Naïve Bayes approach is proved to the most simple and one of the most efficient method of text classification. However one of the drawback of Naiye Bayse classification is that it does not consider the morphological relations of the terms used inside the text. This can be overcome by using other variants of Naïve Bayes algorithms [5]. I. Horie et al., has proposed that the cognitive level identification or difficulty level estimation of the content in text document or in a web document is a relevant research topic today as the number of available materials increased exponentially. A personalized learning material generator for beginner English Learner suggest a model based on the difficulty level by understanding the personalized vocabulary of the learner and then recommends suitable leaning

material to the students. Level of difficulty was determined as a ratio of the number of unknown words in a reading material. The vocabulary level is given to each word which is estimated from Japanese higher general education guide [6]. Atorn Nuntiyagul et al., used patterned keywords and phrase along with support vector machine algorithms and feature selection technique in categorization of items of question stored in an item bank and correctly retrieve the same based on its difficulty levels [7].

A. A. Yahya et.al., analyzed the cognitive levels of class room questions based on Bloom Taxonomy levels. The questions are automatically classified in to different difficulty levels based on Bloom Taxonomy using Machine learning techniques such as K-NN, Naïve Bayses and SVM classifiers using tf as the approach for selection. The results showed that a better performance for the SVM classifiers in terms of accuracy[8]. A. A. Yahya and A Osman constructed different learning paths in e-leaning using Item Response Theory and used Blooms Cognitive levels and support vector machine classification algorithms for question paper classification in automatic text classifications. A set of pre-classified questions are collected and SVM classifier is applied on these data set after preprocessing[9]. K. M. Yang et al., proposed constructing different learning path through e-leaning by analyzing the depth of the user's knowledge on each of the sub topics. The students are grouped together into different levels and identified the knowledge level of each student to learn from the web[10].

II. METHODOLOGY

The most popular e-learning websites for programming languages are used for collecting e-contents after understand its difficulty level by reading through the topics. The websites are categorized in to three based on the difficulty levels level of the e-content such as beginner level intermediate level and advanced level. The web page are collected and parsed and stored as text files in to three different folders after studying the quality and the difficulty level of the e-learning content. The e-contents are collected from multiple e-learning websites such W3schools, geeksforgeeks, roseindia, javaworld, wideskills, oracledocs, java2s, javapoint, guru99, studytonight, edureka, codeside, journeldev, beginnerbook, tutorialpoint, tutorialride, tutorialcup, java2blog, meritcampus, data-flair-training, ibm developer



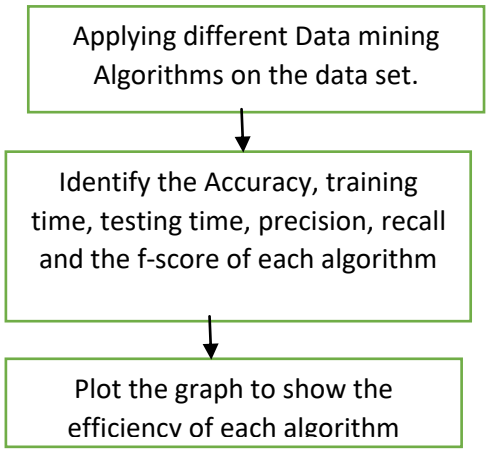


Fig 1: Workflow diagram of the framework proposed framework

The web pages are scraped using scrappy.spider. The Spider crawls the website and downloads the content and text files are extracted using the Beautifulsoup parser API and then stored in local system. The difficulty levels are identified for the content in each of these websites and categorized in to three levels namely- Beginner, Intermediate and Advanced. Data set with different sizes are created to test the accuracy of the classifier at varying volume of data. Text files ranging from 600 to 4000 are used to check the frame work with different data size for further processing.

A. PREPROCESASING

Pre-processing is used to reduce the complexity and increase the accuracy of the data by removing irrelevant and noisy data from the document.

Preprocessing includes removal of unwanted date from the documents. The words in the document will be normalized by converting to any one case and all two letter words are removed. The documents are converted to list of words after removing new line and other metadata elements.

In the pre-processing phase, the downloaded text files are loaded into a python programme in Jupyter Notebook. Preprocessing starts with removing unnecessary words like tabs, punctuations, numeric strings, and quotation marks, single and double letter words and stop words.

The words in the document are normalized by converting to lower case and all two letter words will be removed. The documents are split into number of words through tokenization and converted to list of flat array of words after removing new lines and other metadata elements

B. FEATURE EXTRACTION

Feature extraction uses data reduction which allows the elimination of less important features. The collection of words obtained through tokenization is sorted to find the unique words. The unique words and the count will be stored separately for further processing.

The size of the data is further reduced through the feature selection method. Feature selection reduces the data size, remove redundancy and irrelevant information.

Data reduction is applied by taking different percentage of the total data and applied on the algorithm and check the accuracy of the result.

The data is checked using N-fold cross validation and the best accuracy is obtained when the percentage of the total dimension is taken between 15 to 25. The method help to reduce the dimensionality by 75 to 85% of the total data.

C. CLASSIFICATION ALGORITHMS

The classification algorithms are applied on the data after the pre-processing and dimensionality reduction. The same data is made to run through multiple classification algorithms to compare different classifier performance and to find out the best classier algorithms. The algorithms are compared with different data size, varying training and testing data size and different dimensionality to find out the optimum performance. Different text classification algorithms are trained with training data set and compared the results to find out the best classifier. The classifiers are compared using different data size, different training and testing percentage and different dimensionality.

D. Linear SVC

Support Vector Machine classification implementation using a linear Kernel.is to fit to the data that is provided, returning a best fit hyperplane that divides, or categorizes, data. From there, after getting the hyperplane, can then feed some features to the classifier to see what the "predicted" class. This makes this specific algorithm rather suitable for the use, and the model can be used for many situations[11].

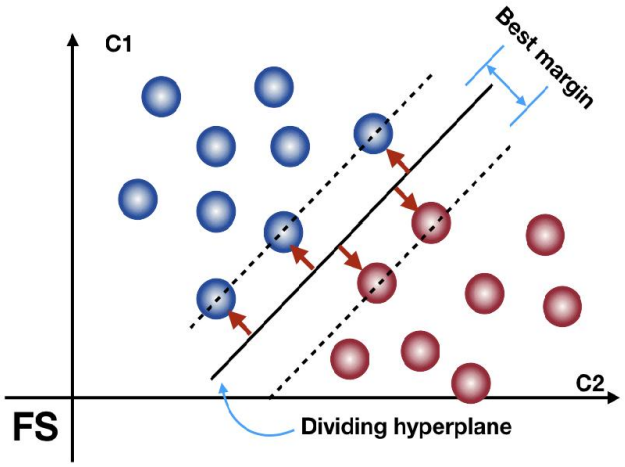


Fig 2: LinerSVC Dividing Hyperplane.

From the figur3 we can understand that the linear function in the middle confirms the best possible separation between the two categories.

In order to predict whether a document D belong to class C if, $\theta^T D \geq 0$. LinearSVC is good when the number of features are high and the number of training examples is small[12]

E. PASSIVE AGGRESSIVE CLASSIFIER

Passive-Aggressive Classifier belong to the family of online learning algorithms for binary classification. It is similar to support vector machine classifier and can be considered as the online version of SVM. It finds a hyperplane to separate the instance into two halves. The margin of an example is proportional to the distance of the hyperplane. The errors in predicting examples can be corrected by using margin of the hyperplane. The update of the classifier follows the constraints passive update and aggressive update.

The advantage is that it follows an online learning pattern and update the separating hyperplane for the next example and ensure the performance of the algorithm[13]. Because of the theoretical loss bound, the performance of the classifier can be easily predicted.

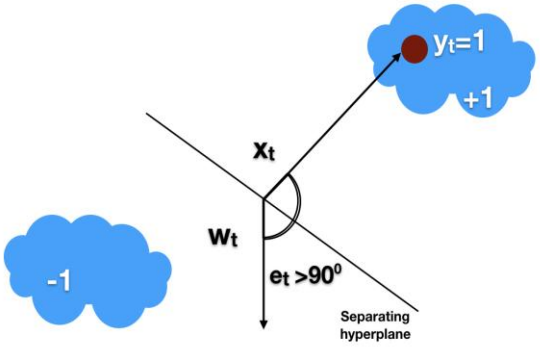


Fig 3: Separating Hyperplane

Figure shows that when the angle Θ>0 the sample is classified as -1 and its label is +1. In such cases the update rule is aggressive and looks for new weight wt for the sample xt.

Passive-aggressive algorithm

INITIALIZE: w1 = (0 . . . 0) as parameters of the classifier

For t = 1, 2…

Receive instance: $\underline{X_t \epsilon R^n}$

Predict $\hat{y}t$= sign (wt-xt)

Receive correct label yt ϵ {-1, +1}

Suffer loss: lt=Max {0, 1- yt(wt- xt)}

Update:

1. Set $\tau t = lt/||xt||2$ (1)
2. Update = wt +1 = wt + $\tau tytxt$ (2)

Where wt is the weight of the vector on round t, yt is the signed margin on round t.

wt+1is set to be the projection of wt in the half-space of vectors that achieve a hinge-loss of zero. The algorithm resulted is passive as the hinge-loss is zero, that is, wt+1=wt when ℓt=0. on those rounds when the loss is positive, the algorithm forces aggressively wt+1 to gratify the constraint ℓ(wt+1;(xt,yt)) =0 irrespective of the size required. Therefore the algorithm is named as Passive-Aggressive[14]

F. RIDGE CLASSIFIER.

Ridge Regression is a technique for analyzing multiple regression data that suffer from multicollinearity. When Multicollinearity occurs, least squares estimates are unbiased, but their variances are large so the values may be far from the true value. By adding a degree of bias to the regression estimates, ridge regression reduces the standard errors.

Multicollinearity, or collinearity, is the existence of near-linear relationships among

The independent variables. Multicollinearity normally occurs due to *Data collection* from narrow substance of the independent Variables, *Physical constraints* of the linear model, *Over-defined model*. There are more variables than observations or *Outliers*

Ridge Classifier uses three phases such as Initialization Phase, Fit Phase, and Predict Phase.

Ridge regression avoids over-fitting by regularizing the weights and keep them small, and model selection is straight forward by choosing the value of a single regression parameter[15].

Following the usual notation, suppose our regression equation is written in matrix form as

$$\underline{Y} = X\underline{B} + \underline{e} \quad (3)$$

Where **Y** is the dependent variable, **X** represents the independent variables, **B** is the regression coefficients to be estimated, and **e** represents residual errors.

The regression coefficients are estimated using the formula

$$\underline{\widehat{B}} = (X'X)^{-1} X'\underline{Y} \quad (4)$$

$$\underline{B'} = (R + kI)^{-1} X'\underline{Y} \quad (5)$$

The estimator is given by,

$$E(\underline{\check{B}} - \underline{B}) = [(X'X + KI)^{-1} \quad X'X - I\,]B \quad (6)$$

The covariance matrix is given by

$$V(\underline{\check{B}}) = [(X'X + KI)^{-1} \quad X'X\,[(X'X + KI)^{-1} \quad (7)$$

Where k is the diagonal element of the cross correlation Metrix and R is a constant

G. STOCHASTIC GRADIENT DECENT CLASSIFIER

In Stochastic Gradient Descent, a few samples are selected randomly instead of the whole data set for each iteration In Gradient Descent, there is a term called "batch" which denotes the total number of samples from a dataset that is used for calculating the gradient for each iteration.Suppose, we have a million samples in our dataset, so if we use a typical Gradient Descent optimization technique, we will have to use all of the one million samples for completing one iteration while performing the Gradient Descent, and it has to be done for every iteration until the minima is reached. Hence, it becomes computationally very expensive to perform. This problem is solved by Stochastic Gradient Descent. In SGD, it uses only a single sample, i.e., a batch size of one, to perform each iteration. The sample is randomly shuffled and selected for performing the iteration

SGD algorithms update the weight vector dynamically. The algorithm[16] study the current point and thus update the weight .SGD is useful for large application with many data-points and problem dimensionality. Optimal performance on large data-point can be obtained by smaller number of iterations over the training data.

H. PERCEPRTON

It takes an input, aggregates it (weighted sum) and returns 1 only if the aggregated sum is more than some threshold else returns 0. The algorithm[17] gives very stable and high reading precision and accuracy level.

$$\text{Y=1 if} \quad \sum_{i=1}^{n} w_i * x_i \geq \theta \quad (8)$$

$$\text{Y=0 if} \quad \sum_{i=1}^{n} w_{i*}\, x_i \leq \theta \quad (9)$$

Rewriting the above

$$\text{Y=1 if} \quad \sum_{i=1}^{n} w_{i*}\, x_i - \theta \geq 0 \quad (10)$$

$$\text{Y=0 if} \quad \sum_{i=1}^{n} w_{i*}\, x_i - \theta \leq 0 \quad (11)$$

Where Xi is the number of conditions added to the perceptron, Wi is the weight of each of the Xi condition. Y is the decision taken by the algorithm based on the condition.

I. K-NEAREST NEIGHBORS ALGORITHEM

K-nearest Neighbors algorithm (K-NN) uses non parametric methods. Input is taken as the k closest training samples from the feature space.

In k-NN classification, object is classified by a group division of its neighbors, the object is assigned to a class most common in its *k* nearest neighbors (The values of *k* is a positive, usually small). If *k* = 1, the object is assigned to the class of single nearest neighbor.

k-NN is a type of instance-based learning, or lazy learning[18] where the function is only approximated locally and all computation is deferred until classification. KNN finds the most similar objects from sample group of objects.

The algorithm usually classifies documents in the Euclidean space as points. Euclidean distance is the distance between two topics in Euclidean space. The distance between two points in the plane with coordinates p= (a, b) and q=(x, y) can be calculated as follows**.**

$$d(p,q) = d(q,p) = \sqrt{(x-a)^2 + (b-y)^2} \quad (12)$$

Where d is the distance between tow pints in the plane, a,b and x,y are coordinates of the two points.

J. NEAREST CENTROID CLASSIFIER.

Nearest centroid classifier is a machine learning classification model that assigns the label of the class of training samples whose centroid is closest to the observation. It is a commonly used method for text classification because of its simplicity. It construct a prototype vector, or centroid, per class using a training set of documents[19].

A test document is assigned to a class that has the most similar centroid [20].Using the cosine similarity measure, a test document can be classified by computing

$$argmax_{1\le i\le p} \frac{q^t ci}{||q||2||ci||2} \quad (13)$$

Where Ci is the centroid of the ith cluster of the training data. When dimension reduction is done by the algorithm, the centroids of the full space become the columns ei ∈ Rp×1 of the identity matrix

K. BernoulliNB

The absence or presence of a word in a text document as feature to classify the document in this model. The words are considered as binary to indicate presence or absence of a word in a document[21]. The model treat a document with set of unique words with no relevance on the frequency of the word.

L. Multinomial NB Classifier

If the number of documents (n) fit into k categories where k ∈ {c1, c2, ..., ck} the predicted class as output is c ∈ C [22]. The algorithm can be written as:

P(c | d) =P(c)∏w∈d P(d | c)nwd/P(d)

Given class c P(w|c) is calculated as

$$p(w|d) = \frac{1 + \sum_{d\in Dc} nwd}{K + \sum_{w'} \sum_{d\in Dc} nw'd} \quad (14)$$

Where nwd is devoted to the number of times word w occurs in document, and P(w|c) is the probability of observing word w.

Random Forest Classifier.

Random forest classifier is built with decision trees. It consist of many individual decision trees functions as an ensemble[23].

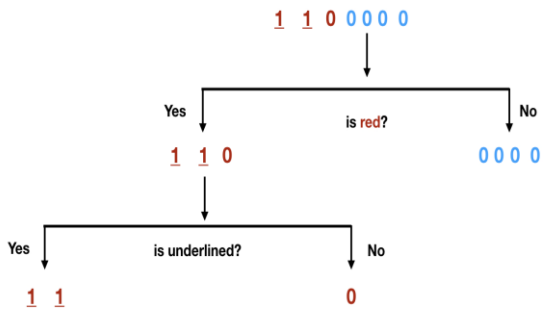


Fig 4: Decision trees.

Figure shows different decision trees in the random forest classifier. Each decision tree gives one output which is either positive or negative. Each tree in the model output a class prediction and the class with the maximum votes is selected as the model prediction. The reason that the random forest model works so well is**,** a large number of relatively uncorrelated models trees function as a group will outperform any of the individual component models. The low relationship between models is the base for achieving the high accuracy. Random forest picks each tree to randomly sample from the dataset with replacement, resulting in many trees. This is known as bagging process.The random forest is a classification algorithm consisting of many decisions trees. It uses bagging and feature randomness when building each individual tree to create an uncorrelated forest of trees whose prediction by group is more precise than that of any individual tree.

IV. IMPLEMENTATION

The accuracy, training and testing time, precision, recall f-score and confusion Metrix are printed and compared with results of different classifiers to predict the accuracy. A Graph is plotted to understand the performance visually. The different classifiers used are RidgeClassifier, Perceptron, Passive Aggressive Classifier, Neighbors Classifier, RandomForestClassifier, LinierSVC Classifier, SGD Classifier, Nearest Centroid Classifier, MultinominalDB, Bernoulli and Pipeline classifier.

With 600 documents with 450 documents for training and 150 for testing and a dimensionality reduction of 85% the below mentioned results are obtained

Table 1 Comparison of different classifiers performance on the dataset

Classifier	Training Time	Testing Time	Accuracy
Ridge Classifier	1.185s	0.109s	0.847
Perceptron	0.187s	0.000s	0.913
Passive-Aggressive	0.109s	0.000s	0.913
KNN	0.062s	0.265s	0.687
Random Forest	0.374s	0.016s	0.987
L2 Penalty (Linear SVC)	0.140s	0.016s	0.9
SGD Classifier	0.187s	0.000s	0.893
L1 Penalty (Linear SVC)	0.094s	0.000s	0.92
Elastic-net Penalty	0.998s	0.000s	0.88
Nearest-Centroid	0.000s	0.016s	0.693
MultinominelNB	0.016s	0.000s	0.887
Bernoulli NB	0.016s	0.016s	0.94
LinearSVC with L1-based feature selection	0.078s	0.000s	0.873

The table show the result of the classifier with dataset of 600 files and the results shows that the Random forest classifier is giving the highest performance followed by Bernoulli NB and LinerSVC.

Table 2 Comparison of different classifiers with precision, recall and f-score.

Classifier	Precision	Recall	F1-Score
Ridge Classifier	0.88	0.86	0.85
Perceptron	0.91	0.91	0.91
Passive-Aggressive	0.91	0.91	0.91
KNN	0.8	0.69	0.69
Random Forest	0.99	0.99	0.99
Linear SVC with L2 Penalty	0.9	0.9	0.9
SGD Classifier	0.89	0.89	0.89
L1 Penalty (Linear SVC)	0.92	0.92	0.92
Elastic-net Penalty	0.88	0.88	0.88
Nearest-Centroid	0.77	0.69	0.7
MultinominelNB	0.89	0.89	0.89
Bernoulli NB	0.94	0.94	0.94
LinearSVC with L1-based feature selection	0.89	0.87	0.87

The table shows the precision, re-call and f-score of different classifiers in the framework. Through the investigation, the current work finds that the random forest classifier is giving the highest precision, recall and f-score followed by Bernoulli NB and Liner SVC L1 Penalty.

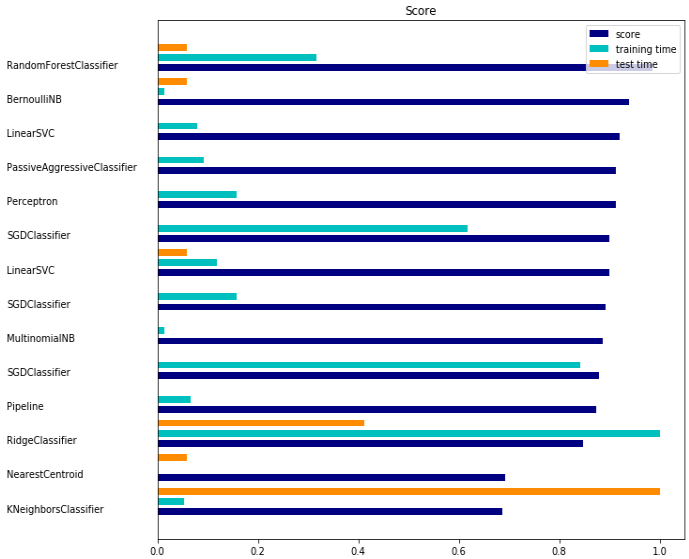


Fig 5: Graphical Representation of the performance of different classifiers with data set size of 600.

When the data size is increased to 4000 documents the results are obtained as below

Table 3 Comparison of different classifiers performance on the dataset with 4000 files

Classifier	Training Time	Test Time	Accuracy
RidgeClassifier	24.500s	0.031s	0.881
Perceptron	2.482s	0.031s	0.928
Passive-Aggressive	5.370s	0.047s	0.944
KNN	1.340s	34.780s	0.696
Random forest	3.198s	0.047s	0.975
LinearSVC-L2 penalty	2.247s	0.033s	0.94
SGDClassifier	5.037s	0.031s	0.924
LinearSVC -L1 penalty	1.871s	0.029s	0.96
SGDClassifier	30.679s	0.031s	0.91
SGDClassifier-Elastic-Net penalty	31.189s	0.031s	0.926
NearestCentroid	0.164s	0.046s	0.726
MultinomialNB	0.138s	0.045s	0.838
BernoulliNB	0.334s	0.094s	0.869
Pipeline	2.233s	0.000s	0.949

Table shows the comparison of different classifiers performance on the dataset with 4000 files. The comparative study made the observation that the random forest classifier is considered as one of the best classifier. Other classification algorithm with good performances are LinearSVC with L1 Penalty. The performance of Naïve base classifiers-BernoulliNB and MultinominelNB reduced on large data size.

Table 4 Comparison of different classifiers performance on the dataset with 4000 files

classifier	Precision	Recall	F1-Score
RidgeClassifier	0.88	0.88	0.879
Perceptron	0.93	0.93	0.93
Passive-Aggressive	0.94	0.94	0.94
kNN	0.73	0.7	0.7
Random forest	0.97	0.97	0.97
LinearSVC-L2 penalty	0.94	0.94	0.94
SGDClassifier	0.92	0.92	0.92
LinearSVC -L1 penalty	0.96	0.96	0.96
SGDClassifier	0.91	0.91	0.91
SGDClassifier-Elastic-Net penalty	0.93	0.93	0.93
NearestCentroid	0.79	0.73	0.74
MultinomialNB	0.86	0.84	0.84
BernoulliNB	0.9	0.87	0.87
Pipeline	0.95	0.95	0.95

The precision recall and f-score of different classification algorithms with dataset of 4000 files are shown in the table. The Random Forest classifier is giving the best result.

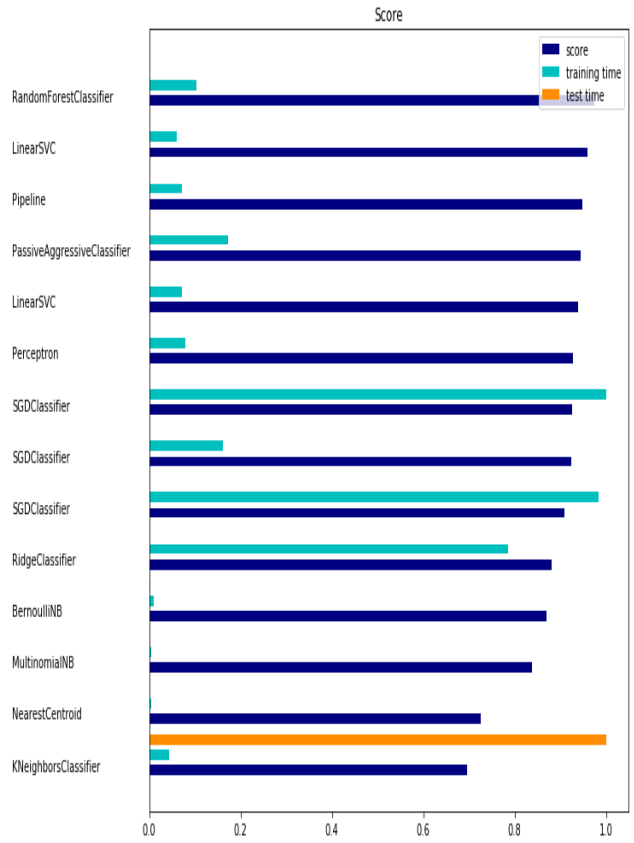


Fig 6: Graphical Representation of the performance of different classifiers with data set (sample size of 600)

It is observed that the random forest classifier is giving the highest precision, recall and f-score followed by and Liner SVC L1 Penalty. The precision recall and f1-score of BernoulliNB and MultinominelNB are considerably reduced on large data size whereas the execution time on training and testing remains low.

V. RESULT AND DISCUSSION

There are many classification algorithms available for classifying unstructured text document. In this work different classifiers are experimentally evaluated with diffret datasets. Through the experimental evaluation, it is observed that the RandomForsetClassifer is giving the highest performance followed by LinaerSVC and Passive aggressive classifier. The naïve base variant BernoulliNB and MultinominelNB are giving good results at small data sets. The performance decreases with large data set. The RandomForset Classifier is selected as the best classifier to develop the model to identify the cognitive level of e-learning contents stored in multiple web sites.

VI. CONCLUSION

As the number of e-learning websites proliferated, it become too difficult for a learner to choose the correct learning content from large collection of web sites. There should be a mechanism to find out the difficulty level of the content to recommend the best e-learning content to a learner based on the knowledge level of the learner. Machine learning classification algorithms can be used for developing a model to classify e-learning content based on its difficulty levels. Through the experimental evaluation and comparison of different classification algorithms, it is found that the Random Forest classifier is giving the best performance and can be used to develop the model frame work for e-leaning content classification.

REFERENCES

1. G. Geetha, M. Safa, C. Fancy, and D. Saranya, "A Hybrid Approach using Collaborative filtering and Content based Filtering for Recommender System," J. Phys. Conf. Ser., vol. 1000, no. 1, 2018.
2. K. I. Ghauth and N. A. Abdullah, "Learning materials recommendation using good learners' ratings and content-based filtering," Educ. Technol. Res. Dev., vol. 58, no. 6, pp. 711–727, 2010.
3. S. Pariserum Perumal, G. Sannasi, and K. Arputharaj, "An intelligent fuzzy rule-based e-learning recommendation system for dynamic user interests," J. Supercomput., no. 0123456789, 2019.
4. J. K. Tarus, Z. Niu, and G. Mustafa, "Knowledge-based recommendation: a review of ontology-based recommender systems for e-learning," Artif. Intell. Rev., vol. 50, no. 1, pp. 21–48, 2018.
5. M. K. M, S. D. H, P. G. Desai, and N. Chiplunkar, "Text Mining Approach to Classify Technical Research Documents using Naïve Bayes," vol. 4, no. 7, pp. 386–391, 2015.
6. I. Horie, K. Yamaguchi, K. Kashiwabara, and Y. Matsuda, "Improvement of difficulty estimation of personalized teaching material generator by JACET," ITHET 2014 - 13th Int. Conf. Inf. Technol. Based High. Educ. Train., 2014.
7. A. Nuntiyagul, K. Naruedomkul, N. Cercone, and D. Wongsawang, "Adaptable learning assistant for item bank management," Comput. Educ., vol. 50, no. 1, pp. 357–370, 2008.
8. A. A. Yahya, A. Osman, A. Taleb, and A. A. Alattab, "Analyzing the Cognitive Level of Classroom Questions Using Machine Learning Techniques," Procedia - Soc. Behav. Sci., vol. 97, pp. 587–595, 2013.
9. A. A. Yahya and A. Osman, "Automatic Classification of Questions Into Bloom's Cognitive Levels Using Support Vector Machine," Proc. Int. Arab Conf. Inf. Technol., no. December 2011, pp. 1–6, 2011.
10. K. M. Yang, R. J. Ross, and S. B. Kim, "Constructing different learning paths through e-learning," Int. Conf. Inf. Technol. Coding Comput. ITCC, vol. 1, pp. 447–452, 2005.
11. F. Amato et al., "Challenge: Processing web texts for classifying job offers," Proc. 2015 IEEE 9th Int. Conf. Semant. Comput. IEEE ICSC 2015, pp. 460–463, 2015.
12. S. Z. Mishu and S. M. Rafiuddin, "Performance analysis of supervised machine learning algorithms for text classification," 19th Int. Conf. Comput. Inf. Technol. ICCIT 2016, pp. 409–413, 2017.
13. H. C. et al., "Comparative Experiments on Sentiment Classification for Online Product Reviews," issue Circ., vol. 115, pp. 1306–1324, 2007.
14. Koby Crammer∗ Ofer, J. Keshet, and S. S.-S. Y. Singer†, "Engineering support for the spine with spondylolisthesis treatment," Online Passiv. Algorithms Koby, vol. 7, pp. 349–354, 2014.
15. A. Singh, B. S. Prakash, and K. Chandrasekaran, "A comparison of linear discriminant analysis and ridge classifier on Twitter data," Proceeding - IEEE Int. Conf. Comput. Commun. Autom. ICCCA 2016, pp. 133–138, 2017.
16. T. Zhang, "Solving large scale linear prediction problems using stochastic gradient descent algorithms," Proceedings, Twenty-First Int. Conf. Mach. Learn. ICML 2004, pp. 919–926, 2004.
17. S. J. Sohrawardi, I. Azam, and S. Hosain, "A comparative study of text classification algorithms on user submitted bug reports," 2014 9th Int. Conf. Digit. Inf. Manag. ICDIM 2014, no. May, pp. 242–247, 2014.
18. B. Trstenjak, S. Mikac, and D. Donko, "KNN with TF-IDF based framework for text categorization," Procedia Eng., vol. 69, pp. 1356–1364, 2014.
19. S. Tan, "An improved centroid classifier for text categorization," Expert Syst. Appl., vol. 35, no. 1–2, pp. 279–285, 2008.
20. H. Kim, P. Rowland, and H. Park, "Dimension reduction in text classification with support vector machines," J. Mach. Learn. Res., vol. 6, pp. 37–53, 2005.
21. C. C. Aggarwal and C. X. Zhai, "A survey of text classification algorithms," Min. Text Data, vol. 9781461432, pp. 163–222, 2012.
22. K. Kowsari, K. J. Meimandi, M. Heidarysafa, S. Mendu, L. Barnes, and D. Brown, "Text classification algorithms: A survey," Inf., vol. 10, no. 4, pp. 1–68, 2019.

23. Q. Wu, Y. Ye, H. Zhang, M. K. Ng, and S. S. Ho, "ForesTexter: An efficient random forest algorithm for imbalanced text categorization," Knowledge-Based Syst., vol. 67, pp. 105–116, 2014.

AUTHORS PROFILE

Mr. Benny Thomas is a research scholar from the Christ (Deemed to be University) Bangalore. He has completed his MPhil in Computer science from Madurai Kamaraj University. He got extensive work experience in industry, as a software developer and trainer. Currently working as assistant professor in the department of computer science, Christ (deemed to be University.) His research interests include Data mining, Big Data Analytics and Deep Learning.

Dr. CHANDRA. J is an Associate Professor in Computer Science Department; CHRIST (Deemed to be University) holds a Masters in Computer Applications from Bharathidasan University. PhD from Hindustan University. Her research interests include Artificial Neural Network, Data Mining, and Genetic algorithm, Big data analytics, Genomics, Deep learning, Convolutional Neural Networks Predictive analytics and Medical Image Processing.

.