International Journal of Innovative Research in Computer Science & Technology (IJIRCST)
ISSN: 2347-5552, Volume-7, Issue-3, May-2019
DOI: 10.21276/ijircst.2019.7.3.6

Smart Document Analysis Using AI-ML

Sindhu Rashmi. H. R, Prof. Anisha. B. S, Dr. Ramakanth Kumar. P

Abstract*— In this era of digitalization, everything is smart and digitalized. All the documents are presented, prepared and shared as soft copies. Classifying those soft copy documents is gaining an important insight in recent times. It is attracting digital word with its impact in different fields like spam filtering, email routing, language identification, genre classification, sentimental analysis, readability assessment. Classifying documents that are available online using smart techniques helps different business. The easiest and efficient way of doing it is through machine learning and it makes human work much easier. To perform classification of document more statistically, documents should be given in a much understandable format to the machine learning classifier. In this report, I'm discussing the types of feature depending on which an document can be classified and later represented. Record arrangement or classifying the documents is the purpose of document collection and classifications based upon the information it consists off and features that it contains. Record arrangement is a huge learning issue that is at the center of numerous data executives and recovery. Document grouping plays an important role in different applications that help with sorting out, ordering, looking and briefly speaking to a lot of data. In this report, we will be discussing the uses of document classification and important steps used for classifying the document or text by considering a small use case to know how document classification is done, basic steps of document classification, processing and analyzing the documents that are collected. We have considered two different categories of data sets for classification and analysis. The problem statement here is to distinguish those two documents where one is Rhyme document and each rhyme is taken as a single file and the other is normal sentences that are a Non-Rhyme document that contains normal Wikipedia text where few statements of Wikipedia is considered as a single file. The precise objective of my project is to develop scalable and efficient document classification project that classifies the document more precisely depending on the feature that it contains and to know the basic techniques that are used for the document a classification like, data collection, data cleaning, pre-processing and constructing an ML model and applying the ML algorithm. Another objective of the project is to work on machine learning concepts and to get insight into different classification algorithms with the help of this case study.

***Index Terms*— ML (Machine Learning), Document classification, Rhyme, Non-Rhyme, Decision Tree Algorithm, Digitalization, Machine Learning Model, Random Forest Algorithm.**

I. INTRODUCTION

In this growing world of information and science, we get everything just with one click and people just browse newspaper, articles, books and read them nobody goes to library search books regarding topics of their interest and study. Nowadays all the pieces of information that we obtain are paperless in another word it is smart. The physical books, papers, notes are replaced completely by smart documents where information is stored and processed [1]. Those documents and files are used for many purposes if they are maintained and processed in a structured way also accessing them would become much easier. Let's take an example of a physical book library where books will be stored in a structured format based on the author, department, subject, specialization, and edition with year of release then searching the book and accessing them will be easier and faster than searching the whole library for single book without having any clue or idea, its time consuming and difficult task. The same way classifying documents becomes difficult to store and becomes more difficult to access when it is needed because of poor classification and arrangement. The undertaking of record arrangement is to dole out a report to at least one classes or classifications. This might be done physically, mentally or algorithmically. The scholarly grouping of archives has for the most part been the area of library science, while the algorithmic characterization of records is predominantly in data science and software engineering. The issues are covering, be that as it may, and there is, accordingly, interdisciplinary research on record order. The records to be arranged might be writings, pictures, music, and so forth. Every sort of record has its extraordinary arrangement issues. At the

Manuscript received May 14, 2019

Sindhu Rashmi. H. R, Department of Software Engineering, RV College of Engineering, Bengaluru, India, 9035383054(sindhu55putani@gamil.com)

Prof. Anisha. B. S, Department of Software Engineering, RV College of Engineering, Bengaluru, India

Dr. Ramakanth Kumar. P, Professor and Head of the Department, Department of CSE, RV College of Engineering Bengaluru, India

point when not generally determined, content characterization is suggested. Archives might be arranged by their subjects or as per different traits, (for example, report type, creator, printing year and so forth.) [2].

Content grouping is utilized to sort out archives in a predefined set of classes. It is extremely valuable in Web content administration, web indexes, email separating, and so forth. Content grouping is a troublesome assignment because of high-dimensional element vector containing boisterous and unimportant highlights. Different component decreases strategies have been proposed for wiping out insignificant highlights just as for diminishing the element of highlight vector. Applicable and diminished element vector is utilized by AI demonstrate for better characterization results. This report presents different content characterization approaches utilizing AI strategies, and highlight determination procedures for decreasing the high-dimensional component vector Mechanized content characterization has been considered as an indispensable technique to oversee and process countless in computerized shapes that are across the board and constantly expanding [3]. All in all, content order assumes a vital job in data extraction and synopsis, content recovery, and question-replying. This work represents the content order process utilizing AI methods. Content order incorporates theme-based content grouping and content type-based characterization. Subject based content arrangement groups archive as indicated by their themes. Writings can likewise be written in numerous classifications, for example: logical articles, news reports, film audits, and commercials. Classification is characterized in transit a content was made, the manner in which it was altered, the register of language it utilizes, and the sort of group of onlookers to whom it is tended to. Past work on sort order perceived that this assignment contrasts from subject based arrangement. Most information for sort grouping is gathered from the web, through newsgroups, announcement sheets, and communicate or printed news. They are multi-source, and thusly have diverse organizations, distinctive favoured vocabularies and frequently altogether extraordinary composition styles notwithstanding for records inside one classification. To be specific, the information are heterogenous [4].

Naturally Text Classification is the undertaking of ordering a report under a predefined classification. All the more formally, on the off chance that I, d is an archive of the whole arrangement of records D and {} 1 2,,, n c is the arrangement of the considerable number of classifications, at that point content characterization appoints one classification j c to a report Id. As in each managed AI task, an underlying dataset is required. A record might be allotted to more than one classification [5][6].

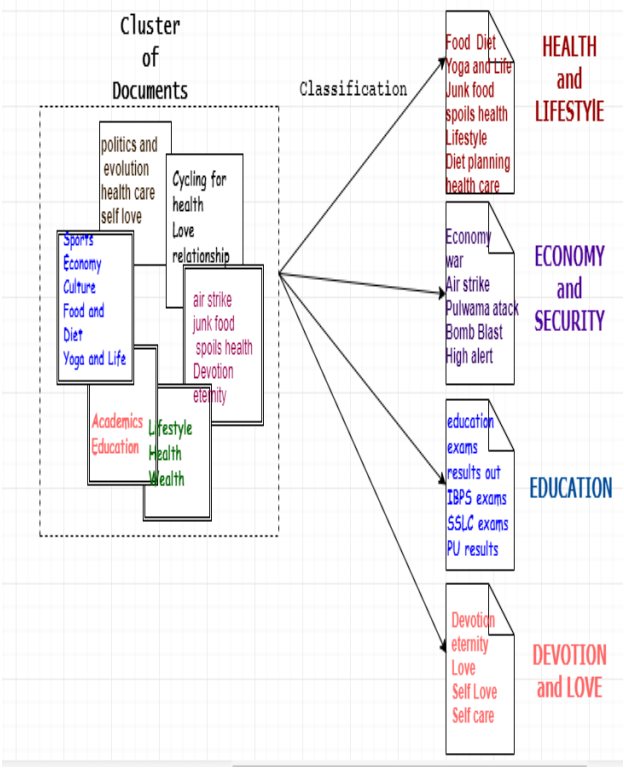

Fig. 1: Pictorial illustration of Document classification.

In this paper we will discuss about Document Classification, collecting and analyzing the data, important and basic steps involved in document classification, all the basic cleaning methods involved in document classification. We'll also understand how the basic ML model is built and how algorithm is applied on that model to find the accuracy.

II. OVERVIEW OF COMPLETE PROJECT DESIGN

Document classification is a grouping of documents as one or more set of predefined type. Document classification targets a large number of messy documents that need to be classified for better understanding and for easy going. Archive grouping is doling out the record to at least one classifications or classes. In this computerized world, everything is changed over to the advanced organization from notes, questions, papers, reading material, books, stories and some more. Report arrangement is utilized in fields, for example, Spam Filtering, Email directing, Language distinguishing proof, Sentimental Analysis, Reliability Analysis [6].

Differentiating between documents is playing a major role in many fields in this era of Digitalization. Nowadays We even get baby rhymes in a smarter way where we can download and use them for teaching children. Hence, I'm using them for my project as a data. Here I'm experimenting this technique by considering two documents one is of Rhyme and the other is a simple Wikipedia statement stored on a file.Here is my project in this I've different phase to collect data, pre-process the collected data, find the feature and decide on the feature to extract, extract the feature, build a model, apply machine learning algorithm, check for the accuracy try new data set on the built model.

International Journal of Innovative Research in Computer Science & Technology (IJIRCST)
ISSN: 2347-5552, Volume-7, Issue-3, May-2019
DOI: 10.21276/ijircst.2019.7.3.6

The first phase is the Collection of Data:

- Here, rhyme data content is taken from an internet source and each rhyme is stored in a single file. As like rhyme data non-rhyme is created by taking simple Wikipedia contents from the internet source. A better and more flexible way to work on data is by converting them to data frames which is nothing but the .csv format. Each text files containing Rhyme data and Non- Rhyme data is converted to data frames (.csv format) with the help of a code snippet where each file is parsed through it and stored as a data frame. Once the data is prepared it is analyzed for extracting the features that help in differentiating between two different documents.
- Depending on the collected data and after analysing the data features from the data need to be decided. Features are the one that helps in differentiating the documents from one another hence one needs to be very care-full upon deciding the features. Here is my project in that, the features that I've considered to differentiate rhyme document from non-rhyme documents are Rhyme count (number of rhyming words), Standard deviation (deviation of lines) and number of words in each line.
- After deciding on the feature by analyzing the data. I wrote a code snippet in python to extract those features where the extracted feature was made to store in a data frame. Because data as data frames are easily handled and used in python and machine learning.
- After extracting the features, a model built depending on the feature and built model is trained by dividing the collected data into test data set and training data set.
- After training the model with the collected set of data, a different algorithms that are specially designed for classification is applied to find the accuracy of the model. I've tried Support Vector Machine and Random Forest where the best accuracy was given by Random forest and I considered the same algorithm to test new data-set. Later new datasets were taken and fed into the model to find its accuracy and to classify the new data.

All the above-specified steps and phases of Document classification using machine learning are overviewed and summarized in the diagram shown below. In the below figure we have two phases one is training phase where raw data is taken, features are extracted by analyzing the data those extracted features are used as an input to the machine learning algorithm to build a classifier model that classifies the documents. The next important phase in my project is the prediction phase. This phase is also known as the testing phase. In this phase, new data is given as an input and the same step takes place where features are extracted from that data and those features are sent to the classifier model that helps in classification. Depending on the extracted feature the classifier model classifies. Finally, after classifying whether it is Rhyme or Nin=Rhyme the output label is given to the outside world.

The input data to the testing phase or prediction phase may be rhyme file or Non-rhyme file. Depending on the features that an input data consists of and depending on the model which was training with some specific features are compared and final prediction is given as an output or final label. The input data to the testing phase or prediction phase may be rhyme file or Non-rhyme file. Depending on the features that an input data consists of and depending on the model which was training with some specific features are compared and final prediction is given as an output or final label.

Smart document classification block diagram and its overview with the functional performance is shown in the blow specified block diagram.

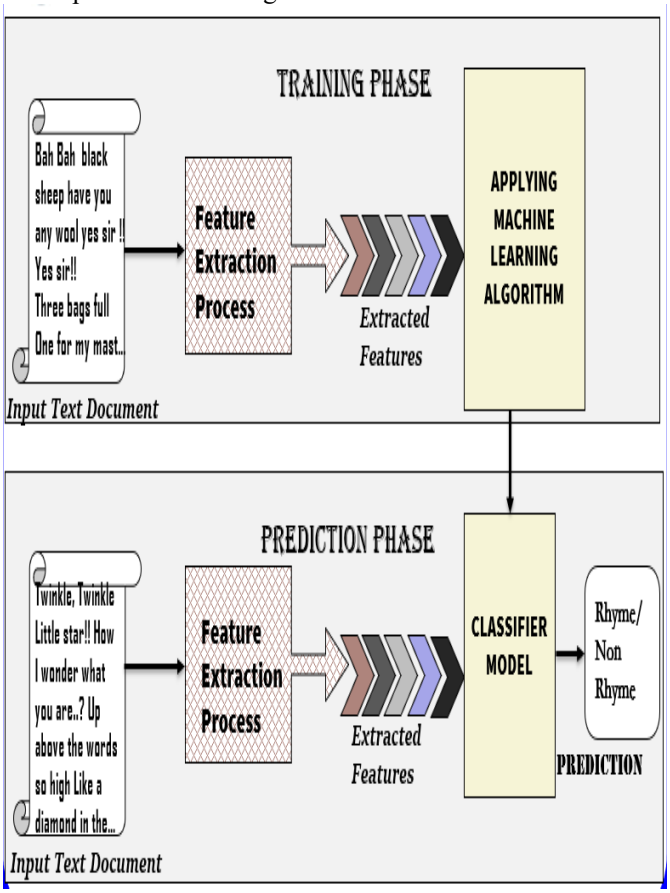

Fig. 2: Architectural Overview of Document Classification using Machine Learning

III. HIGH LEVEL DESIGN

Here in the high-level design with respect to project we have a detailed architecture that explains and outlines the overall structure and overall functionality of the project modules and all the phases that it contains. High-level design of the project explains the overall design of the project with respect to the user. Where one can get complete one time look towards the project from the upper end. The figure 2.1 gives insight towards the project where data is taken as a text document and processed to the desired format depending on the project using Natural Language Processing (NLP). Upon analyzing the data, features are extracted that helps on classification. The desired algorithm is chosen and applied to build an ML model (Machine Learning model). The built model is saved and used for future purpose for testing new data as well.

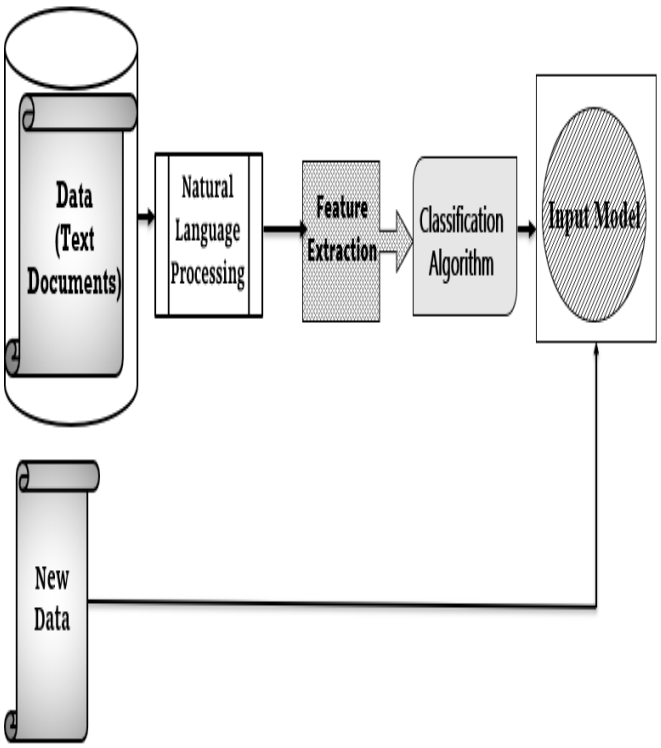

Fig. 3: High-Level Design of Document Classification Using Machine Learning

IV. EXPERIMENTAL DESIGN

Experimental design details about the steps carried out in performing the experiment or the project. It details about the blue print or plan of the project to meet the specified outcome from the project. It explains details of the experiment and it outlines each step of the experiment and all the sub tasks that has been carried out in carrying out the project. Planning the project is very important to reach the specified goal of the project and it also helps in reaching the goal in a specific time constraint. Experimental design also gives a rough outline that when the project can be finished completely when it is designed properly and efficiently. Arranging an analysis legitimately is imperative so as that it guarantee the correct sort of information and efficient example size.

A. Experimental Design 1: Header and Footer Removal

Since each records or text documents od the data may contain headers/footers, for example, From, Subject, date, day, and so on. Disposal of these from the genuine substance of the record can prompt better grouping. It likewise lessens the ideal opportunity for characterization model and accordingly helps in expanding the exactness of the model. This is done easily and efficiently by using Natural Language Processing techniques. But in my datasets which I considered for the experimentation faced less of this problem since I was the one who created the dataset.

B. Experimental Design 2: Stop words Removal

Stop words are the words which occur frequently in the document that has no meaning. Those words won't vomit any meaning in the document, but they are formally called as supporting words for other meaningful words or sentences. These types of words need to be removed in the document for better performance and to save the memory. Practically every one of the records overall classifications contain words like 'it,' 'An,' 'A,' 'THE,' and so on. These words may disturb the task of grouping and may make the grouping results skewed. Evacuation of these words may give progressively precise grouping results.

C. Experimental Design 3: Word Stemming or Stemming of words

Stemming of words is nothing but finding the root word of a specific word and reducing the word size and this methodology also reduce the confusion that occurs between the words since computer doesn't understand any specific language except the binary language. Hence stemming the words reduces the computer confusion and increases the accuracy and efficiency. Example: Think that, we have the accompanying words with recurrence in a given record; walk: 3, strolled: 4, strolling: 6. In this manner as opposed to considering all the three structures independently, we can consider the root word stroll with the recurrence of 13 since all the three words imply a similar significance in an alternate tense. Stemming of words before highlight portrayal can prompt better characterization exactness.

D. Experimental Design 4: Performing Inverse Document Frequency (IDF)

This is one of the important features to be considered in document classification especially. It gives a detail description of number of specific word present in each document. This is a standout amongst the most imperative examinations of the venture work and center of the theory placed previously. The tally vectorizer considers the recurrence of words happening in report regardless of noticeability of words in an archive for highlight portrayal of records. Rather, the archives could be better spoken to on the off chance that we consider the term recurrence alongside what amount recognizing the term is. Such portrayal ought to likewise help improve the general exactness of the model.

E. Experimental Design 5: Feature Extraction

Features are the most important terms that decides the document for which category it belongs. We must be so aware while choosing the features for the project because that is the most important terms to be considered and the model is built depending the proposed or selected feature. Features plays an important role in model building. Analysing the data efficiently and precisely can help in deciding about the features. Features should be decided depending on datasets that has been collected and also final outcome of the project. The features that has been considered for smart document classification using Machine Learning are as follows:

a) Rhyme count
b) Number of words in each line
c) Standard Deviation
d) Defining a Threshold

a) Rhyme count: This is one of the identical and important features to be considered that tells the number of rhyming

ISSN: 2347-5552, Volume-7, Issue-3, May-2019
DOI: 10.21276/ijircst.2019.7.3.6

words present in each line by comparing alternative lines last words in each document. Since the project is all about identifying the rhyme document and non-rhyme document this feature acts a precise part in performing the classification and providing a precise and accurate output. A Python code snippet performs this task and gives the rhyme count that will be stored as a data frame that helps in easy parsing during the model building.

b) Number of words in each line: This is the second important feature that needs to be considered in the project. This feature focuses on several words in each line of a single document. This feature is considered or taken because the rhymes especially baby rhymes that we have considered contains very fewer words in each line which is feasible for babies to learn easily and remember them and also memorize them. But where in a normal Wikipedia text document this isn't the case. It consists of more than 15 words in each line hence these features is considered that yields high accuracy in the model. This feature is performed by a code snippet written in python and extracts the number of words present in each line and writes the data to a corresponding data frame.

c) Standard Deviation: This is another most important feature that is considered for classifying the document as rhyme or non-rhyme. It explains the deviation in the data. The deviation of rhyme data is different from that of non-rhyme data since there exist a smaller number of words in each line of rhyme data where that is completely opposite in non-rhyme data. The standard deviation is the square root of variance. It is given in the formulae

$$\sigma^2 = 1/N \sum (X- \mu)^2$$

where σ^2 is the variance, **N** is the number of observations (whole population), **X** is the individual set of observations and **μ** is the mean.

Standard deviation is a superb method to recognize anomalies. Information focuses that lie beyond what one standard deviation from the mean can be viewed as strange. Much of the time, information focuses that are in excess of two standard deviations from the mean are not considered in the examination. We can discuss how outrageous an information point is by making the inquiry "what number of sigmas from the mean is this".

d) Defining a Threshold: Threshold is nothing but a boundary. Here in this document classification of rhyme or no rhyme documents we are setting a threshold or a boundary on each word. This feature is considered by considering the datasets. Since the datasets are baby rhymes and long sentenced Wikipedia documents, baby rhymes won't be having complex words within it which has many alphabets or characters in it. Since it is a baby rhyme it contains a easy pronounceable small words that contains maximum 7 to 4 character or alphabets within it. Examples: "bah", "bah", "black", "sheep", "twinkle", "little", "star", "Johnny", "yes", "papa". Thus, this feature helps in easy analysis of the document and efficient categorization of the document. Here I have set a threshold of 7 alphabets or characters. A code snippet written in python to find the threshold of each word in a document and overall threshold is considered if the threshold is more than that document can be categorized as Non-Rhyme document.

F. Experimental Design 6: Term Frequency-Inverse Document Frequency (IDF)

The check vectorizer catches more detail than a more straightforward parallel vectorizer, yet it likewise has a specific confinement. In spite of the fact that these considers the recurrence of words happening in a report, it does it regardless of how uncommon or normal the word is. To defeat this impediment Tf-Idf (Term Frequency-Inverse Document Frequency) vectorizer can be utilized. Tf-Idf vectorizer considers the backwards archive recurrence (noticeability weight of the word) alongside the recurrence of each word happening in a report, illuminating the element vector. Give us a chance to state we have a record that contains a word 'get' multiple times and the word 'baseball' multiple times. Here, in the event that we simply utilized term recurrence, we will give more weight to the word 'get' contrasted with the word 'baseball' since it happens all the more much of the time in the report. In any case, the word 'get' may habitually be happening over numerous classes though the word 'baseball' may happen in not many classifications that are identified with games or baseball. In this manner, the word 'baseball' is an all the more distinctive component in the report. In backwards record recurrence, we decide the noticeability of the word which we then increase with term recurrence to get the new weight of each word in the archive. How the Tf-Idf is determined is appeared as follows:

TF (Term Frequency) = Number of times term/word happens in the record.

IDF (Inverse Document Frequency) = log (N/1 + {d ϵ D: t ϵ d})

Here, N is an absolute number of records in the corpus and {d ϵ D: t ϵ d} is various archives where term t shows up. Tf-Idf is determined as:

Tf-Idf = TF * IDF

Accordingly, if that word occurs less time over various reports than its noticeability or in progressively specialized terms it's IDF esteem will be high and the word that much of the time happens crosswise over numerous reports will have a low IDF esteem. In this manner, in Tf-Idf the element vector won't be founded exclusively on term recurrence of words yet will be a result of term recurrence alongside its IDF esteem.

V. ALGORITHMIC EXPLANATION

The algorithm that has been used in the project is Random Forest algorithm. Which is well known as efficient classification algorithm? In which classification algorithm is the one that gives output as "0" or '1" that is, "TRUE" or "FALSE" and hence this speciality of the algorithm makes me to choose this algorithm for the project Smart Document Analysis Using Machine Learning where the project was to classify the document as "Rhyme" or "Non-Rhyme". Thus, the algorithm was considered for the project to classify two

different set of documents in a smarter way. Arbitrary Forest is a versatile, easy to use AI count that produces, even without hyper-parameter tuning, a mind-boggling result as a rule. It is moreover a champion among the most used counts since its straightforwardness and the way that it might be used for both request and backslide assignments. Let us now understand the detailed design of the algorithm, feature of the algorithm and also let's discuss what makes the algorithm special when compared to other different set of algorithms in this chapter which is titled as algorithmic explanation [10].

A. Working of Random Forest Algorithm

Random forest is the most famous algorithm that is used for classification which is used as both classification and regression algorithm also it is a combination of decision tree algorithm where many decision trees together forms a random forest. Many decision trees combine to produce a large bag of trees and those classification result of each decision trees are considered and compared with other decision trees result and maximum output from many decision trees are considered and maximum outputs from the decision tree's is considered as final output of the random forest algorithm. It is also called as Arbitrary Forest. As we understood from its name, it makes backwoods and uses different strategies of implementation. Forest in this algorithm mainly termed as a group of Trees more precisely decision trees, which are not prepared with technique of packing. Common use of packing technique is nothing but it trains learning models to builds the general way of outputs.[11] [12].

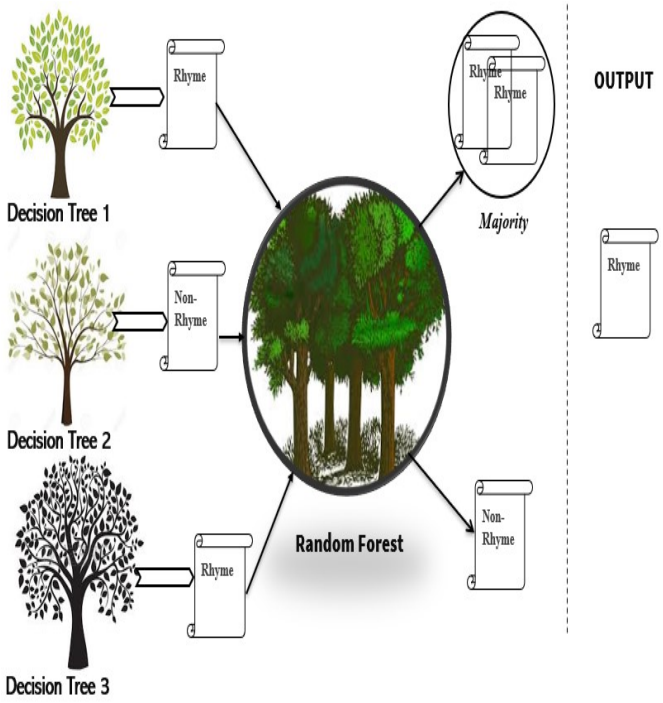

Fig. 4: Pictorial representation of Random Forest Algorithm

Random Forest has almost a similar hyperparameters as choice tree or classifier packing. It is not necessary to consolidate a tree choice with a packing classifier we just use a classifier that is, Random Forest. Random Forest manages Regression by utilizing the regressor of Random Forest. Irregular Forest implements more haphazardness to the model during development of tree. Than hunting down the most important or different component, it searches for the best of best element among an irregular subset. These outputs a wide variety in a superior model. In addition to these, Random Forest is an irregular subset defined by the calculation for a defined part. We can also make trees increasingly irregular, by using arbitrary edges for specific component hunting down the most identical edges [14].

B. Vital Hypermeter in Random Forest Algorithm

The Hyperparameters in the irregular backwoods used to increase the intensity prescient of the model or to make the model quicker. I will be discussing here the hyperparameters of sklearns worked for the project.

a. Predictive Power Expanding

Directly off the bat, there is the "n_estimators" hyperparameter, which is just the number of trees the estimation works before taking the most outrageous throwing a ticket or taking midpoints of conjectures. When in doubt, a higher number of trees grows the execution and makes the desires continuously consistent, anyway it in like manner impedes the count [15]. Another basic hyperparameter is "max_features", which is the most extraordinary number of features Random Forest thinks about separating a centre point. Sklearn gives a couple of options, portrayed in their documentation. The last fundamental hyper-parameter we will talk about in regard to speed is "min_sample_leaf" [16]. This chooses, like its name starting at now says, the base number of leaves that are required to section an inside center point [17] [18].

b. Expanding the Models Speed

The Hyperparameters in the sporadic boondocks are either used to extend the farsighted power of the model or to make the model snappier. I will here talk about the hyperparameters of scholarly work in subjective forest work [19]. All in all, there is the "oob_score" which is a self-assertive timberland cross-endorsement system. In this testing, around 33% of the data isn't used to set up the model and can be used to survey its execution. These precedents are known as the out of sack tests. It is in a general sense equivalent to the overlook one cross-endorsement procedure, be that as it may, no additional computational weight obliges it [20] [21].

c. Preferences and Disadvantages of Random Forest

Like I recently referenced, the influence of arbitrary timberland is that it might be used for both relapse and request errands or grouping and that it's definitely not hard to see the relative centrality it doles out to the data features [23]. Unpredictable Forest is similarly considered as an incredibly supportive and easy to use the figuring since it's default hyperparameters every now and again produce a better than average desire result. The amount of hyperparameters is in like manner not too much high and they are clear to get it. One of the huge issues in AI is overfitting, yet as a general rule, this won't happen that easy to a subjective timberland classifier. That is in such a case, that there are adequate trees in the forest, the classifier won't overfit the model [24] [25]. The crucial hindrance of Random Forest is that a considerable number of trees can

ISSN: 2347-5552, Volume-7, Issue-3, May-2019
DOI: 10.21276/ijircst.2019.7.3.6

impact the count to move back to and inadequate for consistent desires. All things considered, these computations rush to plan yet direct to make desires once they are readied. A dynamically accurate estimate requires more trees, which results is moderate. In most obvious applications the Random Forest computation is speedy enough, be that as it may, there can totally be conditions where run-time execution is basic and various approaches would be favoured. Additionally, clearly, Random Forest is a perceptive exhibiting instrument and not an unmistakable gadget. That infers, in case you are hunting down a portrayal of the associations in your data, various procedures would be favoured.

VI. APPROACHES AND METHOD

Approach and methods of Smart Document Classification using Machine Learning explain the detailed implementation steps of the project. It also focuses on the approach that is used in the project also forecast the methods on which it is carried out. It explains the project implementation in a defined format and details out the code that was used to implement the specific feature in the project. It also explains the flow of the project how it was carried out and steps taken to carry out the flow. In this chapter, we will see the technical steps and technical implementation that was considered and carried out from the beginning until the end of the project. Since this project was done innovatively thinking from the scratch that is, from building dataset by own till applying the algorithm on the model and getting the result. It was a challenging process and the complete process and procedure of the project and its technical details are detailed below in this chapter.

A. Gathering Datasets

Since the project was new and nobody has tried doing this project of classifying the two document where one in Rhyme and another one is non-rhyme I had to face few problem in the beginning to get the data set and initially I did not get any rhyme dataset and non-rhyme dataset anywhere on the internet and hence I started constructing my own dataset by browsing about the baby rhymes and collecting those baby rhymes by copying each rhyme into a single text file and saved them in a different folder and then coming to non-rhyme files I browsed some Wikipedia information about different topics and saved 20-25 lines of the Wikipedia information as one file and saved those file as a different folder. Rhyme folder has more than 1000 rhyme dataset that contained unique rhymes and Non-Rhyme folder had more than 2000 non-rhyme dataset that contained unique Wikipedia statements. The below figure gives the pictorial representation of the raw data set how it looked in the initial stage of collection. Where few rhyme files are shown below:

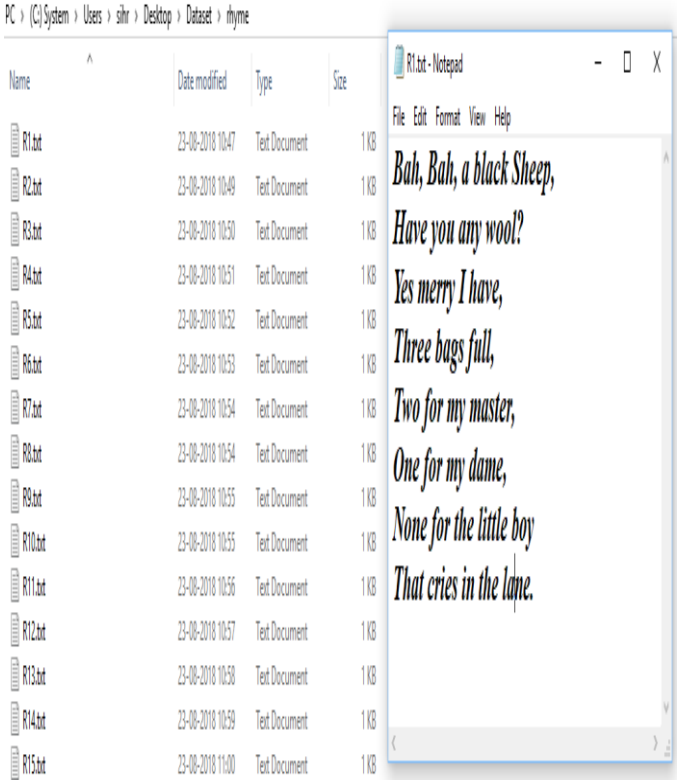

Fig. 5: Pictorial representation of the raw datasets collected (Rhyme dataset).

The pictorial representation of the raw dataset collected for non-rhyme dataset is picturised below.

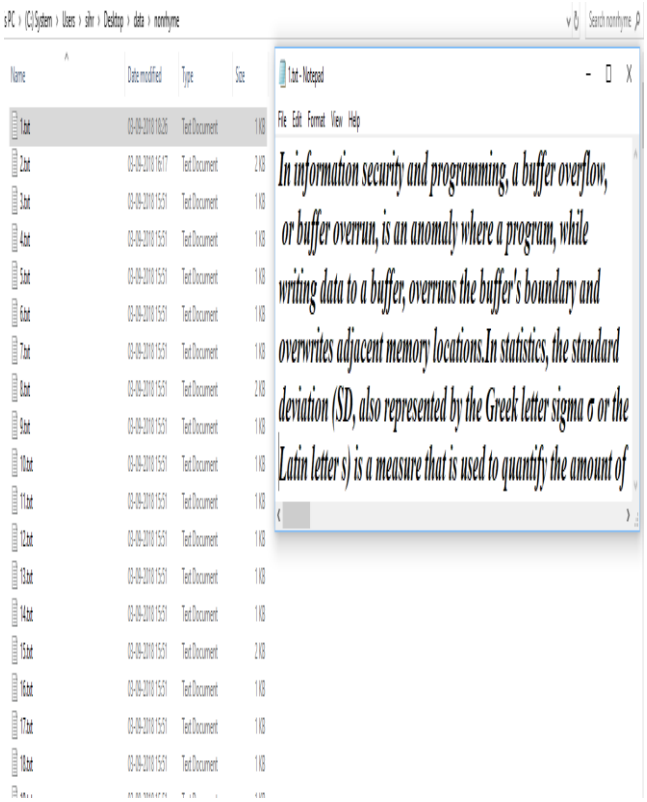

Fig. 6: Pictorial representation of the raw datasets collected (Non-Rhyme dataset).

The above-given datasets show how the data when collected raw looked like. Now cleaning the dataset in this form was not a feasible step to take. To clean the data set and to give the data a definite structure I thought of converting them to a data frame where the dataset is stored in an XL sheet in row and column. When the data are in this format then it is easily be fetched, altered, removed or deleted and hence that helps cleaning technique easy, feasible and reliable. A code snippet was written to convert the data in a folder which 1000+ files into a data frame. The pictorial representation of a structured data is given below

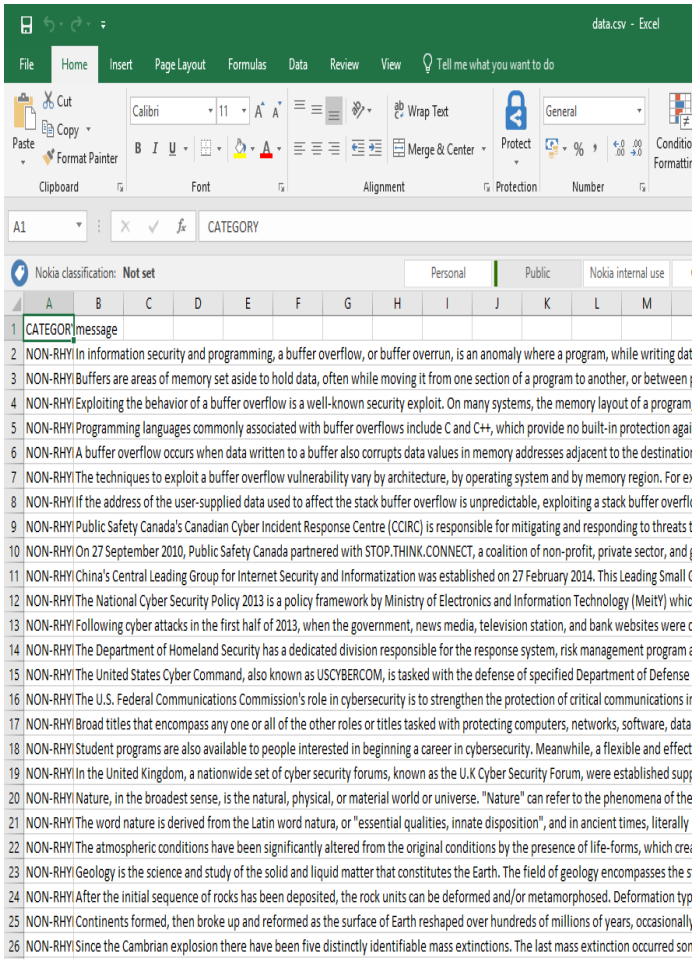

Fig. 7: Pictorial representation of structured dataset as data frames(.CSV) format

Code snippet used to perform the conversion of collected raw dataset into structured data frame is screenshotted as below.

```
import os
import pandas as pd

def read_file(filename):
    with open(filename,'r',encoding='iso-8859-15') as out:
        lines = out.readlines()
        return lines
data = []
path = r'''C:/Users/sihr/Desktop/Data'''
for folder in sorted(os.listdir(path)):
    for file in sorted(os.listdir(path+'/'+folder)):
                pathname = path+'/'+folder+'/'+file
                file_read = read_file(pathname)
                data.append((folder, file_read))
```

Fig. 8: Code snippet to convert raw datasets to structured data frames

This code snippet in the above figure 5.2.4 takes the collected data files from the folder one by one and converts them into data frames and stores in row and column format. Loading the data is another important task to be considered where informational collections(datasets) are now in CSV records that are information outlines contains the comma isolated values, which can be stacked effectively, the errand of stacking the informational collection was somewhat tangled. The informational index contained two essential envelopes named as 'Rhyme' and 'Non-Rhyme.' Each organizer further contained 1500+ records, one for every class of archives and they are stacked into the model that is built later. where the dataset was separated as testing and training dataset used to an efficient model.

B. Data Cleaning

Once the data collected is converted to a structured format that is, data frames and loaded, then it is easy to proceed on the dataset. We should initially understand the dataset how it is stored and what is there in it to know what is necessarily and what is unnecessary in the dataset to proceed with the project by applying cleaning techniques upon the dataset to obtain the higher accuracy in the classification. Data set and the cleaning techniques that we take plays an important role in deciding the output and accuracy of the project. One should focus on it An introductory cleaning system that happens in any archive order is expelling headers, and footers of the archive. These may incorporate 'From,' 'Subject,' 'Association,' 'Telephone,' 'Fax' and so forth. Evacuation of these abandons us with the genuine substance of the report and the class to which it has a place. This encourages us to constrain the length of vocabulary (however still immense). Additionally, these headers and footers don't contribute in any huge manner in helping us accomplish our target that is the arrangement of archives dependent on the genuine substance of records. mAfter removing the common cleaning technique for all the document classification that is, headers and footers. Now one must concentrate on cleaning technique that is related to the project. Type of cleaning technique we apply completely depends on the type of dataset that we have chosen and type of model we are building. There are many different cleaning techniques available but choosing a precise one depending on the

International Journal of Innovative Research in Computer Science & Technology (IJIRCST)
ISSN: 2347-5552, Volume-7, Issue-3, May-2019
DOI: 10.21276/ijircst.2019.7.3.6

dataset and the project we are doing in another important task of each machine learning engineer.

The different cleaning techniques used in the project are:

a. **Vocabulary generation**
b. Stop words removal
c. Stemming

a. Vocabulary generation: Subsequent step after cleaning of the dataset, is the production of vocabulary. Vocabulary is set or union of everything that has occurred once in a document or dataset. To more readily comprehend the idea, think about a precedent given here. Let me take chance to state two reports S1 and S2. Where S1 contains,

S1 = {Bah, bah, black sheep, Have you any wool? Yes, sir, yes, sir, three bags full; One for the master, and one for the dame, and one for the little boy Who lives down the lane.}

S2 = {Bah, Bah, black Sheep, Have you any wool? Yes, merry I have, three bags full, two for my master, one for my dame, None for the little boy That cries in the lane.}

Here the vocabulary V is nothing but a union of S1 and S2 that contains all the words that are available in both the datasets at least once within the documents. This is the self-prepared vocabulary by considering the input data from the user. The vocabulary produced from the above explained set of documents is given below:
V={'bah' , 'Sheep', 'black' 'have', 'any', 'wool', 'you' 'yes', 'sir', 'three', 'bags', 'full', 'one', 'for', 'the', 'master', 'dame, 'and', 'little', 'boy', 'who', 'lives', 'down ', 'lane', 'merry', 'I', 'two', 'none', 'that', cries', 'in')

b. Stop words removal
The issue with this sort of Vocabulary is that it contains many stop words. Stop words are the regular English words that don't help in the characterization of archives by any means. The stop words just connect two or more meaningful words or sentences Give me a chance to break down the Vocabulary we just made. It as of now contains plenty of stop words. Let us see the stop words that our vocabulary contains that was created recently above [27].
SW = {'for', 'the', 'and', 'who', 'in', 'that')

These Stop Words (SW) has no connectivity between the two set of classification or two documents and it just increases the size of the dataset and makes it clumsy and it becomes difficult for classification when these stop words occur many a times in all the documents that's been considered. Also, these words have no connection with both two classifications and may show up in all classifications whose reports are under scrutiny. In this way, in Vocabulary creation, these words will be evacuated. The stop word evacuated vocabulary (SWRV) will contain every one of the words that happen at any rate once in the preparation set of reports aside from the stop words. For our precedent, the stop words expelled vocabulary will look like as appeared as follows. The stop words evacuated dictionary for the above vocabulary that we obtained from two set S1 and S2 is as follows:

SWRV = {'bah', 'black', 'Sheep', 'have', 'you', 'any', 'wool', 'yes', 'sir', 'three', 'bags', 'full', 'one', 'master', 'dame, 'little', 'boy', 'lives', 'down ', 'lane', 'merry', 'I', 'two', 'none', 'cries')

c. Performing stemming of words:
In spite of the fact that SWRV (Stop Word Removed Vocabulary) will work superior to the less complex vocabulary V, despite everything it can be improved by utilizing Stemming. Stemming is the way toward changing over bent or inflected (changed structure) words to their stem words. Consider, for our precedent we get another archive S3.
S3 = {'As', 'I', 'was', 'going', 'to', 'St', 'Ives', 'I', 'met', 'a', 'man', 'with', 'seven', 'wives', 'Each', 'wife', 'had', 'seven', 'sacks', 'Each', 'sack', 'had', 'seven', 'cats', 'Each', 'cat', 'had', 'seven', 'kits', 'Kits', 'cats', 'sacks', 'and', 'wives', 'How', 'many', 'were', 'there', 'going', 'to', 'Ives'}

Here S3 contains the word "had" which is a root word of "have" where the word "have" is already in the vocabulary that has been generated. vocabulary V contains the word which is its root word. Hence the word "had" is not considered and it is not updated in the vocabulary when reading S3. Since both the word passes on a similar significance and originate from a similar root word, it doesn't bode well to keep both in the vocabulary. This expansion to the vocabulary could have been maintained a strategic distance from on the off chance that we had performed stemming. Stemming not simply help in restricting the extent of the Vocabulary yet additionally helps in keeping the span of the component vector under tight restraints

```
import re
from nltk.corpus import stopwords
REPLACE_BY_SPACE_RE = re.compile('[/(){}\[\]\|@,;]')
BAD_SYMBOLS_RE = re.compile('[^0-9a-z #+_]')
STOPWORDS = set(stopwords.words('english'))



from bs4 import BeautifulSoup
def text_cleaning(text):
    """
        text: a string


        return: modified initial string
    """
    #text = BeautifulSoup(text, "lxml").text # HTML decoding
    text = text.lower() # lowercase text
    text = REPLACE_BY_SPACE_RE.sub(' ', text) # replace REPLACE_BY_SPACE_RE symbols by space in text
    text = BAD_SYMBOLS_RE.sub('', text) # delete symbols which are in BAD_SYMBOLS_RE from text
    text = ' '.join(word for word in text.split() if word not in STOPWORDS) # delete stopwors from text
    text = ' '.join(word for word in text.split() if word.isalpha())



    return text
```

Fig. 9: Code snippet that performs data cleaning Process

The above code snippet performs all the data cleaning process to give a more feasible dataset to build an efficient model. It also helps in building a more feasible Machine Learning model which can also be used for other projects cleaning process where they can alter the code according to their needs and availability [28].

C. Feature Extraction and Representation

Features are the most important terms that decides the document for which category it belongs. We must be so aware while choosing the features for the project because that is the most important terms to be considered and the model is built depending the proposed or selected feature. Features plays an important role in model building. Analysing the data efficiently and precisely can help in deciding about the features. Features should be decided depending on datasets that has been collected and also final outcome of the project. The features that has been considered for smart document classification using Machine Learning are as follows:

a. Rhyme count
b. Number of words in each line
c. Standard Deviation
d. Defining a Threshold

a. Rhyme count: This is one of the identical and important features to be considered that tells the number of rhyming words present in each line by comparing alternative lines last words in each document. Since the project is all about identifying the rhyme document and non-rhyme document this feature plays an important role in performing the classification and providing a precise and accurate output. A Python code snippet performs this task and gives the rhyme count that will be stored as a data frame that helps in easy parsing during the model building. The figure below represents the technique for getting the rhyme count of the document

```
def function_last_count(list_match):
    count = 0
    list_match=data['text']
    arpabet=nltk.corpus.cmudict.dict()
    for idx,word in enumerate(list_match):
        length = len(list_match)
        if(idx < length-1):
            word1 = list_match[idx]
            word2 = list_match[idx+1]
            if ((word1) in arpabet) & ((word2) in arpabet):
                wordlist_1 =    arpabet[word1]
                wordlist_2 =    arpabet[word2]

                if (len(wordlist_1) & len(wordlist_2)):
                    word_list_1_last = wordlist_1[0][-1]
                    word_list_2_last = wordlist_2[0][-1]
                    if word_list_1_last == word_list_2_last:
                        count = count+1

                if((len(wordlist_1) >=2 ) & (len(wordlist_2) >= 2)):
                    word_list_1_second = wordlist_1[0][-2]
                    word_list_2_second = wordlist_2[0][-2]
                    if word_list_1_second == word_list_2_second:
                        count = count+1


    return count
```

Fig. 10: Code snippet for applying a Rhyme count feature

After applying the above rhyme count feature on the dataset then we get the number of rhyming words from each alternate line's in the document.

b. Number of words in each line: This is the second important feature that needs to be considered in the project. This feature focuses on several words in each line of a single document. This feature is considered or taken because the rhymes especially baby rhymes that we have considered contains very fewer words in each line which is feasible for babies to learn easily and remember them and also memorize them. But, where in a normal Wikipedia text document this isn't the case. It consists of more than 15 words in each line hence these features are considered that yields high accuracy in the model. This feature is performed by a code snippet written in python and extracts the number of words present in each line and writes the data to a corresponding data frame. The figure below represents the wordcount of the document.

International Journal of Innovative Research in Computer Science & Technology (IJIRCST)
ISSN: 2347-5552, Volume-7, Issue-3, May-2019
DOI: 10.21276/ijircst.2019.7.3.6

```
def func_count(data):
    wordCounter = {}
    i=0
    line = data['text']
    word_list = ''.join([item.rstrip('\n') for item in line])
    word_list = ''.join([item.rstrip(',') for item in line])
    word_list = ''.join([item.rstrip('/') for item in line])
    word_list = word_list.lower().split()
#     word_list = line.replace(',','').lower().split()
#     word_list = [w.replace('/', '') for w in word_list]
#     word_list = [w.replace('.', '') for w in word_list]
#     ''.join([item.rstrip('\n') for item in gerritinfo])
    for word in word_list:

            if word not in wordCounter:
              wordCounter[word] = 1
            else:

              wordCounter[word] = wordCounter[word] + 1



    for  (word,count)  in wordCounter.items():
            if count>1:
              i=i+1

    return i
```

Fig. 11: Code snippet for applying a wordcount feature

After applying on the data frames the output of the code snippet is shown below:

```
dataframe['count'] = dataframe.apply(func_count,axis=1)
dataframe
```

	output	text	count
0	rhyme	[Bah, Bah, a black Sheep,\n, Have you any wool...	4
1	rhyme	[Can you count the stars that brightly\n, twin...	10
2	rhyme	[Lucy Locket lost her pocket,\n, Kitty Fisher ...	3
3	rhyme	[Mary had a little lamb, little lamb,\n, littl...	11
4	rhyme	[He followed her to school one day,\n, school ...	14
5	rhyme	[And so the teacher turned it out,\n, turned i...	12
6	rhyme	["Why does the lamb love Mary so\n, love Mary ...	12
7	rhyme	[Mary, Mary, quite contrary,\n, How does your ...	2
8	rhyme	[Mistress Mary, Quite contrary,\n, How does yo...	2
9	rhyme	[Mistress Mary, Quite contrary,\n, How does yo...	0
10	rhyme	[Cowslips all in arow\n, With lady bells all i...	2
11	rhyme	[Matthew, Mark, Luke and John,\n, Bless the be...	5
12	rhyme	[Cock a doodle do!\n, What is my dame to do?\n...	11
13	rhyme	[Matthew, Mark, Luke and John,\n, The Bed be b...	0
14	rhyme	[Four newkS in this house, for haly Angels,\n,...	3
15	rhyme	[Matthew, Mark, Luke, and John,\n, Bless the b...	2
16	rhyme	[White Pater-noster, St Peter□s brother,\n, Wh...	9
17	rhyme	[White paternoster,\n, God was my Foster.\n, S...	11
18	rhyme	[Monday's child is fair of face\n, Tuesday's c...	6
19	rhyme	[Needles and pins,\n, Needles and pins,\n, Whe...	3

Fig. 12: Output representation after applying a feature on the data frames

c. Standard Deviation: This is another most important feature that is considered for classifying the document as rhyme or non-rhyme. It explains the deviation in the data. The deviation of rhyme data is different from that of non-rhyme data since there exist a smaller number of words in each line of rhyme data where that is completely opposite in non-rhyme data. The standard deviation is the square root of variance. It is given in the formulae

$$\sigma^2 = 1/N \sum (X- \mu)^2$$

where σ^2 is the variance, N is the number of observations (whole population), X is the individual set of observations and μ is the mean [29].

Standard deviation is a superb method to recognize anomalies. Information focuses that lie beyond what one standard deviation from the mean can be viewed as strange. Much of the time, information focuses that are in excess of two standard deviations from the mean are not considered in the examination. We can discuss how outrageous an information point is by making the inquiry "what number of sigmas from the mean is this" [30] [31].

```
def func_sd(data):
    line = data['text']
    import statistics
    lines = 0
    words = 0
    twords=0
    b=[]
    characters = 0
    for line in line:
        wordslist = line.split()
        lines = lines + 1
        words = words + len(wordslist)
        twords = twords + len(wordslist)
        characters = characters + len(line)
        b.append(words)
        words=0


    for c in b:
        if c == 0:
            b.remove(c)

    e=statistics.stdev(b)
    return e
```

Fig. 13: Code snippet for applying a standard deviation feature

After applying on the data frames the output of the code snippet is shown below:

```
dataframe['standarddeviation'] = dataframe.apply(func_sd,axis=1)
```

```
dataframe
```

	output	text	count	threashold	standarddeviation
0	rhyme	[Bah, Bah, a black Sheep,\n, Have you any wool...	4	0	0.707107
1	rhyme	[Can you count the stars that brightly\n, twin...	10	4	0.834523
2	rhyme	[Lucy Locket lost her pocket,\n, Kitty Fisher ...	3	0	0.816497
3	rhyme	[Mary had a little lamb, little lamb,\n, littl...	11	2	1.397276
4	rhyme	[He followed her to school one day,\n, school ...	14	4	0.744024
5	rhyme	[And so the teacher turned it out,\n, turned i...	12	3	1.356203
6	rhyme	["Why does the lamb love Mary so\n, love Mary ...	12	1	1.309307
7	rhyme	[Mary, Mary, quite contrary,\n, How does your ...	2	1	1.290994
8	rhyme	[Mistress Mary, Quite contrary,\n, How does yo...	2	2	0.816497
9	rhyme	[Mistress Mary, Quite contrary,\n, How does yo...	0	3	0.957427
10	rhyme	[Cowslips all in arow\n, With lady bells all i...	2	1	2.121320
11	rhyme	[Matthew, Mark, Luke and John,\n, Bless the be...	5	1	1.095445
12	rhyme	[Cock a doodle do!\n, What is my dame to do?\n...	11	4	0.755929
13	rhyme	[Matthew, Mark, Luke and John,\n, The Bed be b...	0	1	2.121320
14	rhyme	[Four newkS in this house, for haly Angels,\n,...	3	2	2.516611
15	rhyme	[Matthew, Mark, Luke, and John,\n, Bless the b...	2	2	1.500000
16	rhyme	[White Pater-noster, St Peter□s brother,\n, Wh...	9	4	2.714160
17	rhyme	[White paternoster,\n, God was my Foster.\n, S...	11	3	1.548366
18	rhyme	[Monday's child is fair of face\n, Tuesday's c...	6	6	1.457738
19	rhyme	[Needles and pins,\n, Needles and pins,\n, Whe...	3	1	0.500000
r199416370	68c087924	down to sleep,\n, I pray the Lor...	7	0	0.577350

Fig. 14: Output representation after applying a feature on the data frames

d. Defining a Threshold: Threshold is nothing but a boundary. Here in this document classification of rhyme or no rhyme documents we are setting a threshold or a boundary on each word. This feature is considered by considering the datasets. Since the datasets are baby rhymes and long sentenced Wikipedia documents, baby rhymes won't be having complex words within it which has many alphabets or characters in it. Since it is a baby rhyme it contains a easy pronounceable small words that contains maximum 7 to 4 character or alphabets within it. Examples: "bah", "bah", "black", "sheep", "twinkle", "little", "star", "Johnny", "yes", "papa". Thus, this feature helps in easy analysis of the document and efficient categorization of the document. Here I have set a threshold of 7 alphabets or

International Journal of Innovative Research in Computer Science & Technology (IJIRCST)
ISSN: 2347-5552, Volume-7, Issue-3, May-2019
DOI: 10.21276/ijircst.2019.7.3.6

characters. A code snippet written in python to find the threshold of each word in a document and overall threshold is considered if the threshold is more than that document can be categorized as Non-Rhyme document.

```
def func_threashold(data):
    line = data['text']

    j=0
    b=[]
    word_list = ''.join([item.rstrip('\n') for item in line])
    word_list = ''.join([item.rstrip(',') for item in line])
    word_list = ''.join([item.rstrip('/') for item in line])
    word_list = word_list.lower().split()

    for x in [len(x) for x in word_list]:

            if x>7:
                j=j+1
    return j
```

Fig. 15: Code snippet to obtain threshold feature on the documents

The above functional code is applied on each data frame by using the pandas that is, dataframe.apply (function name, axis=1). The below line of code.

```
dataframe['threashold'] = dataframe.apply(func_threashold,axis=1)
```

Fig. 16: Code snippet to apply threshold feature on the documents

After applying the feature extraction function that is, threshold to each individual data frame the output is again stored corresponding to the specific data frame in the .CSV format so that it is easy to build a model and apply algorithm to the data frames. The output of the feature is shown below in the figure 7.4.4(c).

```
dataframe
```

	output	text	count	threashold	standarddeviation
0	rhyme	[Bah, Bah, a black Sheep,\n, Have you any wool...	4	0	0.707107
1	rhyme	[Can you count the stars that brightly\n, twin...	10	4	0.834523
2	rhyme	[Lucy Locket lost her pocket,\n, Kitty Fisher ...	3	0	0.816497
3	rhyme	[Mary had a little lamb, little lamb,\n, littl...	11	2	1.397276
4	rhyme	[He followed her to school one day,\n, school ...	14	4	0.744024
5	rhyme	[And so the teacher turned it out,\n, turned i...	12	3	1.356203
6	rhyme	["Why does the lamb love Mary so\n, love Mary ...	12	1	1.309307
7	rhyme	[Mary, Mary, quite contrary,\n, How does your ...	2	1	1.290994
8	rhyme	[Mistress Mary, Quite contrary,\n, How does yo...	2	2	0.816497
9	rhyme	[Mistress Mary, Quite contrary,\n, How does yo...	0	3	0.957427
10	rhyme	[Cowslips all in arow\n, With lady bells all i...	2	1	2.121320
11	rhyme	[Matthew, Mark, Luke and John,\n, Bless the be...	5	1	1.095445
12	rhyme	[Cock a doodle do!\n, What is my dame to do?\n...	11	4	0.755929
13	rhyme	[Matthew, Mark, Luke and John,\n, The Bed be b...	0	1	2.121320
14	rhyme	[Four newkS in this house, for haly Angels,\n,...	3	2	2.516611
15	rhyme	[Matthew, Mark, Luke, and John,\n, Bless the b...	2	2	1.500000
16	rhyme	[White Pater-noster, St Peter□s brother,\n, Wh...	9	4	2.714160
17	rhyme	[White paternoster,\n, God was my Foster.\n, S...	11	3	1.548366
18	rhyme	[Monday's child is fair of face\n, Tuesday's c...	6	6	1.457738

Fig. 17: Output representation after applying a feature on the data frames.

e. Training the model

After deciding on the features and extracting those features from the dataset then the next step is building a model (Machine Learning model) with the extracted feature.The model is built by considering all the features that has been extracted in the above steps. That is, word count, rhyme count, standard deviation, threshold all those columns including the labels (rhyme label, Non-rhyme label) in which datasets are labelled depending on the data that it holds also data taken. Later model is built by considering the collected dataset and dividing those datasets into training test and test set. The percentage of splitting test and training test is 30:70. That is, the percentage of data considered for training is 70% and percentage of data considered for testing is 30%. The training par's code snippet is picturised in below. This code snippet explains the training part of the model with the help of data and the feature it holds.

```
dataframe.to_csv('rhyme.csv',header = True , index = False)
```

```
rhyme_dataset = pd.read_csv("C:/Users/sihr/final.csv")
```

```
rhyme_dataset.columns
```

```
Index(['Label', 'Message', 'count', 'threashold', 'standarddeviation'], dtype='object')
```

```
from sklearn.model_selection import train_test_split
```

```
droplist = ['Label','Message']
X = rhyme_dataset.drop(droplist,axis=1)
y = rhyme_dataset['Label']
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.30, random_state=101)
```

Fig. 18: code snippet that showcases the model building phase.

f. Applying Algorithm

After developing a Machine Learning model, the next and final step of the project is to apply suitable algorithm upon the model and check for the accuracy and discuss the result. After checking the model with many algorithms, I settled finally for "Random Forest Algorithm" that gave me higher accuracy of 99%. The algorithm fitted the model in such a way that it gave only one miss classification and 99 correct classification [31]. Random forest is basically famous algorithm for classification and it is well known as classification algorithm. I opted random forest algorithm for my model because my project is completely a classification project where the model was built to classify two different documentation one is rhyme and another is non-rhyme and hence I looked for more précised classification algorithm, after doing trial and error method on many classification algorithm like decision tree, Naïve Bayes and Support Vector Machine I finally opted more efficient, suitable and precise algorithm for my project that is, Random forest algorithm [31] [32]. The following figure depicts the application of Random Forest Algorithm on the previously built model.

```
from sklearn.ensemble import RandomForestClassifier
```

```
C:\Users\nselvan\AppData\Local\Continuum\anaconda3\lib\site-packages\sklea
g: numpy.core.umath_tests is an internal NumPy module and should not be im
e.
  from numpy.core.umath_tests import inner1d
```

```
rfc = RandomForestClassifier(n_estimators=600)
```

```
rfc.fit(X_train,y_train)
```

```
RandomForestClassifier(bootstrap=True, class_weight=None, criterion='gini',
            max_depth=None, max_features='auto', max_leaf_nodes=None,
            min_impurity_decrease=0.0, min_impurity_split=None,
            min_samples_leaf=1, min_samples_split=2,
            min_weight_fraction_leaf=0.0, n_estimators=600, n_jobs=1,
            oob_score=False, random_state=None, verbose=0,
            warm_start=False)
```

```
predictions = rfc.predict(X_test)
```

```
from sklearn.metrics import classification_report,confusion_matrix
```

```
print(classification_report(y_test,predictions))
```

```
              precision    recall  f1-score   support

    nonrhyme       0.99      1.00      1.00       150
       rhyme       1.00      0.99      1.00       161

 avg / total       1.00      1.00      1.00       311
```

```
print(confusion_matrix(y_test,predictions))
```

```
[[150   0]
 [  1 160]]
```

Fig. 19: Code snippet to show the algorithmic implementation of the project

The above figure shows the algorithmic implementation of Random Forest algorithm on the pre-built machine learning model by considering all the features that had been decided previously depending on the aim of the project and depending on the dataset that was collected. After applying Random Forest algorithm on the project, it yielded 99% of accuracy with one misclassification.

VIII. RESULT AND DISCUSSION

The data for the project Smart Document Analysis Using Machine Learning was collected from the scratch using the data available on the internet and also from the nursery rhyme books. Rhyme data was collected from online available rhymes on the internet and 100+ rhymes were typed by looking at the nursery rhyme's book and non-rhyme data was collected completely from the internet with the help of Wikipedia information. Each file size was 107kb to 500 kb. Features were selected by analysing the data and the model was built. later Random Forest algorithm applied on the model and tested for accuracy

where it yields 99% accuracy which is a remarkable one. Now, coming to features that was selected for the project Of course, Tf-Idf vectorizer beats the twofold vectorizer and the check vectorizer. Its output's 5.2% preferred exactness over the vectorizer check and 7.3% preferred precision over the double vectorizer. It also covered a superior exactness score of 0.79 contrasted with the check vectorizer and the parallel vectorizer where both got the better Precision score of 0.69. It overcame rest two vectorizers in Recall score also getting a Recall score of 0.69 over 0.79 of the check vectorizer and 0.79 of the parallel vectorizer. F-1 Score is determined dependent on Recall and Precision, it came better for vectorizer Tf-Idf. Numerically Tf-Idf vectorizer accomplished F-1 score of 0.87 beating 0.74 of the paired vectorizer and 0.72 vectorizer of the check. As represented in the report, better performance was given by Tf-Idf vectorizer. Because it uses noticeability factor of every component in gauging contrasted with simply the recurrence of terms in the document. This makes performance of Tf-Idf better and exact manner. One intriguing perception from the outcomes is that if stop words in not evacuated at that point, double vectorizer shows improvement over check vectorizer. Vectorizer of Parallel obtained 63% compared with 59% vectorizer of tally. This is because it contains more stop words. Along these lines, if term recurrence speaks to a report, at that point the stop words are probably going to turn into the most critical component of the record. We realize that stop words are never great separating criteria, it results in misclassifications, prompting a drop in the precision of forecasts [33]. The figure below shows the result of the model when tested with Random forest algorithm. The result is depicted as a confusion matrix where only one was a misclassification rest was correct prediction.

```
rfc = RandomForestClassifier(n_estimators=600)
```

```
rfc.fit(X_train,y_train)
```

```
RandomForestClassifier(bootstrap=True, class_weight=None, criterion='gini',
            max_depth=None, max_features='auto', max_leaf_nodes=None,
            min_impurity_decrease=0.0, min_impurity_split=None,
            min_samples_leaf=1, min_samples_split=2,
            min_weight_fraction_leaf=0.0, n_estimators=600, n_jobs=1,
            oob_score=False, random_state=None, verbose=0,
            warm_start=False)
```

```
predictions = rfc.predict(X_test)
```

```
from sklearn.metrics import classification_report,confusion_matrix
```

```
print(classification_report(y_test,predictions))
```

```
              precision    recall  f1-score   support

    nonrhyme       0.99      1.00      1.00       150
       rhyme       1.00      0.99      1.00       161

 avg / total       1.00      1.00      1.00       311
```

Fig. 20: Result analysis: confusion matrix that depicts the accuracy of the model

IX. CONCLUSION AND FUTURE ENHANCEMENT

The project Smart Document Classification Using Machine Learning is an unique project which can be considered as a demo project to understand the basics of Data collection, Data analysis, data preparation, data pre-processing by applying different types of cleaning technique it also gives idea on the Machine learning models, feature extraction process, Machine learning algorithm for the researchers who are keenly interested in the field of data science. This project can be used by researchers to look at an introductory or sample project to know how the classification of document works and how features are classified and basic steps that need to be considered while trying to classify the documents. This project also helps learners to understand machine learning basics also data cleaning techniques. It acts as a demo project or large documentation projects. One can easily understand the basics of the smart document classification and how machine learning can help in making document classification smarter. Here, I had used only one algorithmic technique, and that is only the Random Forest calculation for grouping as the fundamental point of our task work was to break down the distinctive sorts of highlight portrayal for reports. Additionally, as future work, I might want to take the work forward to attempt and test the distinctive component portrayal plans and furthermore attempting the model with other AI calculations like SVM, Neural Network, Expedition amplification, Decision trees, and so forth. I likewise attempted another vectorizer that is, Hashing Vectorizer. The hashing vectorizer despite the fact that did not execute just as Tf-Idf vectorizer but rather was essentially quicker than all the vectorizer portrayal referenced in the report. It doesn't require the vocabulary to be available in memory constantly, and in this way, it is space effective also. In the event that the scientist needs to do archive characterization progressively, I would recommend them to investigate hashing vectorizer.

X. REFERENCES

[1] Ankit Basarkar "Document classification using Machine Learning [1]", Springer International Conference 5-25-2017 Vol 531 © 2018.

[2] Berina Alic, Lejila Gurbeta and Almir Badnjevic, "Machine Learning Techniques for Classification of Diabetes and Cardiovascular Diseases" 2017, 6th MEDITERRANEAN CONFERENCE ON EMBEDDED COMPUTING," (MECO), 11-15 JUNE 2017, BAR, MONTENEGRO, 978-1-5090-6742-8/17/$31.00 ©2017 IEEE

[3] Zhongmin Luo, "CDS Rate Construction Methods by Machine learning Techniques", Social Science Research Network (SSRN) Electronic journal on May 12 2107.

[4] Suresh Yaram "Machine Learning Algorithms for Document clustering and Fraud Detection" 2016 IEEE International Conference on Data Science and Engineering (ICDSE) 978-1-5090-1281-7/16/$31.00 ©2016 IEEE

[5] Leila Arras, Franziska Horn, Gregoire Montavon, Klaus-Robert Muller, "What is Relevant in a Text Document?" An Interpretable Machine Learning Approach, International Workshop on Analytics and Networking arXiv:1612.07843v1 [cs.CL] 23 Dec 2016

[6] Arthi Venkataraman, "Deep Learning Algorithms Based Text Classifier", 2nd International Conference on Applied and Theoretical Computing and Communication Technology (iCATccT), 978-1-5090-2399-8/16/$31.00_c 2016 IEEE

[7] V. V. Gulin and A. B. Frolov, "On the Classification of Text Documents Taking into Account Their Structural Features" Journal of Computer and Systems Sciences International, 2016, Vol. 55, No. 3, pp. 394–403. © Pleiades Publishing, Ltd., ISSN 1064_2307, 2016.

[8] P. O. Lima Junior, L. G. Castro Junior and A. L. Zambalde, "Analysis of Machine Learning Tecniques to Classify News for Information Management in Coffee Market", International Conference on Digitalization, IEEE LATIN AMERICA TRANSACTIONS, VOL. 13, NO. 7, JULY 2015

[9] Siwei Lai, Liheng Xu, Kang Liu and Jun Zhao, "Recurreny Convolution Neural Networks for Text Classification", Proceedings of the Twenty-Ninth AAAI Conference on Artificial Intelligence, Association for the Advancement of Artificial Intelligence on 2015.

[10] Liang Yang and Hongfei Lin, "C.nstruction and Application of Chinese Emotional Corpus", Chinese Lexical Semantics Workshop (CLSW) 2013, LNAI 7717, pp. 1–10, 2015. © Springer-Verlag Berlin Heidelberg 2015.

[11] Marenglen Biba and Mersida Mane, "Sentiment Analysis through Machine Learning: An Experimental Evaluation for Albanian", Recent Advances in Intelligent Informatics, Advances in Intelligent Systems and Computing 235,195 DOI: 10.1007/978-3-319-01778-5_20, © Springer International Publishing Switzerland 2014.

[12] Bina Kotiyal, Ankit Kumar, Bhaskar Pant and R. H. Goudr, "Classification Technique for Improving User Acces on Web Log Data", International conference on Intelligent Computing, Networking and Informatics, Online ISBN978-81-322-1665-0 on 18 December 2014

[13] Maofu Liu, Yu Xiao, Chunwei Lei and Xin Zhou, "Social Relation Extraction Based on Chinese Wikipedia Articles", Chinese Lexical Semantics Workshop (CLSW) 2014, LNAI 7717, pp. 1–10, 2013. © Springer-Verlag Berlin Heidelberg 2014.

[14] B. S. Harish and B. Udayasri, "Document Classification: An Approach Using Feature Clustering", IEEE Conference on Recent Advances in Intelligent Informatics, Advances in Intelligent Systems and Computing 235, DOI: 10.1007/978-3-319-01778 5_17, © Springer International Publishing Switzerland 2014

[15] Guo-Nian Wang, Yi Qin, Mini Jiang, Qiu-Rong Zhao, "MT-Oriented and Computer- Based Subject Restoration for Chinese Empty-Subject Sentences", Chinese Lexical Semantics Workshop (CLSW) 2013, LNAI 7717, pp. 1–10, 2013. © Springer-Verlag Berlin Heidelberg 2013

[16]Muhammad Shahbaz, Qanta Ahmed and Aziz Guergachi, "An Expert Framework For Effective Document Classifictaion Using Support Vector Machine", International Journal of Innovative Computing Information and Control ICIC International Conference, Volume 9, Number 4, April 2013 ©2013 ISSN 1349-4198.

[17] Yonglei Zhang, Cheng Peng and Hongling Wang, "Research on Chinese Sentence Compression for the Titke Generation", Chinese Lexical Semantics Workshop (CLSW) 2013, LNAI 7717, pp. 1–10, 2013. © Springer-Verlag Berlin Heidelberg 2013

[18] Shengfeng ju and Shoushan Li, "Active Learning in Sentiment Classification by Selecting Both Words and Documents", Chinese Lexical Semantics Workshop (CLSW) 2013, LNAI 7717, pp. 1–10, 2013. © Springer-Verlag Berlin Heidelberg 2013.

[19] Xiuli Hua, Shoushan Li, Peifeng Li and Qiaoming Zhu, Reseach on Intrinsic Plagiarism Detection Resolution: A supervised Learning Approach", Chinese Lexical Semantics Workshop (CLSW) 2013, LNAI 7717, pp. 1–10, 2013. © Springer-Verlag Berlin Heidelberg 2013.

[20] Zhu Zhu, Daming Dai, Yaxing Ding, Jianbin Qian and Shoushan Li, "Employing Emotion Keywords to Improve Cross-Domain Sentiment Classification", Chinese Lexical Semantics Workshop (CLSW) 2013, LNAI 7717, pp. 1–10, 2013. © Springer-Verlag Berlin Heidelberg 2013.

[21] Ge Xu, Chu-Ren Huang and Houfeng Wang, "Extracting Chinese Product Features: Representing a Sequence by a Set of Skip-Bigrams", Chinese Lexical Semantics Workshop (CLSW) 2013, LNAI 7717, pp. 1–10, 2013. © Springer-Verlag Berlin Heidelberg 2013.

[22] Charles Smutz and Angelos Stavrou, "Malicious PDF Detection using Metadata and Structural Features", *Annual Computer Security Applications Conference* (*ACSAC*) 2012 *ACSAC* '12 Dec. 3-7, 2012, Orlando, Florida USA, 2012 ACM 978-1-4503-1312-4/12/12 ...$15.00.

[23] Gerhard Paass and Luliu Konya, "Machine Learning for Document Structure Recognition", Studies in Computational Intelligence on June 22nd, 2011.

[24] Jyri Saarikoski, Jorma Laurikkala, Kalervo Jarvelin and Martti Juhola, "Self-Organizing Maps in Document Classification: A Comparision with Six Machine Learning Methods", Internation Conference on Adaptive and Natural Computing Algorithms (ICANNGA) 2011, Part I, LNCS 6593, pp. 260–269, 2011. © Springer-Verlag Berlin Heidelberg 2011

[25] Bhawna Nigam, Poorvi Ahirwal, Sonal Salve, Swati Vamney, "Document Classification Using Expectation Maximization with Semi Supervised Learning", International Journal on Soft Computing (IJSC) Vol.2, No.4, DOI: 10.5121/ijsc.2011.2404 November 2011.

[26] Dilara Torunoglu, Erhan Cakirman, Murat Can Ganiz, et.al, "Analysis of Processing Methods on Classification of Turkish Texts", International

Conference on Informational Technology with Machine Learning" 978-1-61284-5/11/$26.00 ©2011 IEEE

[27] Yu Wanjun and Song Xiaoguang, "Research on Text Categorization Based on Machine Learning" IEEE International Journal on Machine Learning and its Implementation, 978-1-4244-6932-1/10/$26.00 ©2010 IEEE

[28] R. Deepa Lakshmi and N.Radha, "Spam Classification using Supervised Learning Techniques", International Conference onWomen in Applied Computing and Information Technology. A2CWiC 2010, September 16-17, 2010, India Copyright © 2010 978-1-4503-0194-7/10/0009… $10.00

[29] Baharum Baharudin, Khairullah khan, Lam Hong Lee, Aurangzeb Khan, "A Review of Machine Learning Algorithms for Text-Document Classification", JOURNAL OF ADVANCES IN INFORMATION TECHNOLOGY, VOL. 1, NO. 1, FEBRUARY 2010 Published on 2010

[30] Janusz Wnek, "Machine Learning of Document Templates for Data Extraction", U. S. Conference on Science and Application, U.S. Patent, US 7,764,830 B1, July 27, 2010

[31] Aurangzeb Khan, Baharum Baharudin, Lam Hong Lee, Khairullah Khan, "A Review of Machine Learning Algorithms for Text-Documents Classification", JOURNAL OF ADVANCES IN INFORMATION TECHNOLOGY, VOL. 1, NO. 1, FEBRUARY 2010 © 2010 ACADEMY PUBLISHER doi:10.4304/jait.1.1.4-20

[32] Simon Tong and Daphne Koller, "Support Vector Machine Active Learning with Applications to Text Classification", Journal of Machine Learning Research 2010 on 11/01/2010

[33] Konstantin Mertsalov and Michael McCreary, "Document Classification with Support Vector Machines", International Conference on IEEE Transactions on Knowledge and Data Engineering on January 2009.

