Journal of King Saud University – Computer and Information Sciences (2016) **28**, 330–344

King Saud University

Journal of King Saud University – Computer and Information Sciences

www.ksu.edu.sa
www.sciencedirect.com

Sentiment classification of Roman-Urdu opinions using Naïve Bayesian, Decision Tree and KNN classification techniques

Muhammad Bilal *, **Huma Israr, Muhammad Shahid, Amin Khan**

CS/IT Department, IBMS, University of Agriculture, Peshawar, Pakistan

Received 31 July 2015; revised 14 October 2015; accepted 4 November 2015
Available online 12 December 2015

KEYWORDS

Roman Urdu;
Opinion mining;
Bag of words;
Naïve Bayes;
Decision Tree;
k-Nearest Neighbor

Abstract Sentiment mining is a field of text mining to determine the attitude of people about a particular product, topic, politician in newsgroup posts, review sites, comments on facebook posts twitter, etc. There are many issues involved in opinion mining. One important issue is that opinions could be in different languages (English, Urdu, Arabic, etc.). To tackle each language according to its orientation is a challenging task. Most of the research work in sentiment mining has been done in English language. Currently, limited research is being carried out on sentiment classification of other languages like Arabic, Italian, Urdu and Hindi. In this paper, three classification models are used for text classification using Waikato Environment for Knowledge Analysis (WEKA). Opinions written in Roman-Urdu and English are extracted from a blog. These extracted opinions are documented in text files to prepare a training dataset containing 150 positive and 150 negative opinions, as labeled examples. Testing data set is supplied to three different models and the results in each case are analyzed. The results show that Naïve Bayesian outperformed Decision Tree and KNN in terms of more accuracy, precision, recall and F-measure.

1. Introduction

Due to extensive use of computers, smartphones and high speed internet, people are now using web for social contacts, business correspondence, e-marketing, e-commerce, e-surveys, etc. People share their ideas, suggestions, comments and opinions about a particular product, service, political entity and current affairs. There are so many user-generated opinions available on the web. From all those opinions, it is difficult to judge the number of positive and negative opinions (Khushboo et al., 2012). It makes it difficult for people to take the right decision about purchasing a particular product. On the

* Corresponding author.
E-mail addresses: qec_mbilal@aup.edu.pk (M. Bilal), huma.israr@gmail.com (H. Israr), shahid_swabi@yahoo.com (M. Shahid), amin-khan@aup.edu.pk (A. Khan).
Peer review under responsibility of King Saud University.

ELSEVIER Production and hosting by Elsevier

http://dx.doi.org/10.1016/j.jksuci.2015.11.003
1319-1578

other hand, it is also difficult for manufacturers or service providers to keep the track of the public opinions about their products or service and to manage the opinions. Similarly, an analyst wants to conduct a survey to get feedback of public on a specific topic. He/She will post the topic on a blog to analyze the sentiment of people about that topic. There will be so many opinions on that post. For all these opinions, it will be difficult to know how many opinions are positive and negative. So a computer machine may be trained to take such decisions in a quick and accurate manner.

The important thing in opinion mining is to extract and analyze the feedback of people in order to discover their sentiments. Growing availability of opinion-rich resources like online blogs, social media, review sites; raised new opportunities and challenges (Pang and Lee, 2008). People now can actively use information technologies to search the opinions of others.

There are many issues involved in opinion mining. The first is some words in opinion are representing a positive sense in one situation and negative in the other. For example consider an opinion "the size of this mobile is **small**". Here the word **small** comes in positive sense. On other hand, consider another opinion, "The battery time of this mobile is **small**". Here the word **small** is interpreted negatively (Rashid et al., 2013). Another issue in opinion mining is that most of the text processing system depends on the fact that a small difference in two sentences does not change the meaning very much. In sentiment analysis, the text "the movie was great" is different from "the movie was not great". People may have contradiction in their statements. Most of the reviews have both positive and negative comments, which is a bit manageable by analyzing sentences one at a time. However in more informal medium like facebook, twitter and blogs, lack of context makes it difficult for the people to understand what someone thought based on a short piece of text. One important issue in opinion mining is that product reviews, comments and feedback could be in different languages (English, Urdu, Arabic, etc.), therefore to tackle each language according to its orientation is a challenging task (Rashid et al., 2013).

Most of the research work in sentiment mining has been done in English and Chinese languages. Currently, limited research is conducted on sentiment classification for other languages like Arabic, Italian, Urdu and Hindi, etc. Urdu is an Indo-Aryan language which uses extended Persian and Arabic script. Roman script for Urdu does not have any standard for the spelling of the word. A word can be written in different forms with different spellings not only by distinct people but also by the same person at different occasions. Specially, there is no one to one mapping between Urdu letters for vowel sounds and the corresponding roman letters (Ahmed, 2009). There is no major difference in the pronunciation of Urdu and Hindi, therefore the roman version of Urdu and Hindi are written almost the same. Hence, this research is conducted in Roman Urdu and could be applicable in Roman Hindi. These are the most spoken languages in Pakistan, India, Bangladesh and among the people of these areas living in different parts of the world.

Previous work (Daud et al., 2014) conducted Roman Urdu opinion mining by using the key matching method. Adjectives of the opinions were matched with a manually designed dictionary to find polarity of that opinion. It was found that the accuracy of that work was low because the adjective alone cannot determine the polarity of an opinion. For example, consider a comment "I really **like** Iphone" here adjective is **Like** which has positive sense but on the other hand, consider another comment "I didn't **like** Iphone" here adjective is again **Like** which gives a positive sense but the comment interprets negative sentiments about Iphone. So it shows that all words of the opinions are equally important to indicate a comment either positive or negative. Thus the proposed model will use Bag of Words Model and three different classification techniques to improve the accuracy of Roman-Urdu sentiment classification.

The objectives of this research are to mine the polarity of public opinions written in Roman-Urdu with blend of English and Urdu extracted from a blog, to train the machine using a training data set, and to build Naïve Bayesian, Decision Tree and KNN classification models and to predict the polarity of new opinions by using these classification models.

This paper is organized into five sections. In the first and second sections the introduction and previous related work is briefly described. In the third section, the methodology adopted to perform different experiments is explained. In the fourth section, calculation and evaluation of experiments are performed to get various results and discussion on these results is conducted. In the last section, certain conclusions are drawn on the basis of outcomes.

2. Related work

In 2015, Daud et al. proposed a system called Roman Urdu Opinion Mining System (RUoMiS) which uses natural language processing technique to find the polarity of the opinion. In this study, the adjectives in the opinions were compared with a manually designed dictionary to find the polarity of the opinions. The results of the experiment were recorded with a precision of 27.1%, however, RUoMiS categorized about 21.1% opinions falsely. In 2014, Kaur et al. used a hybrid technique for Punjabi text classification (Kaur et al., 2014). In this research the combination of Naïve Bayesian and N-gram techniques were used. The features of the N-gram model were extracted and then used as training dataset to train Naïve Bayes. The model was then tested by supplying testing data. It was found that by comparing results from already existing methods, the accuracy of the proposed method was effective. Ashari et al. in 2013, used Naïve Bayes, Decision Tree, and *k*-Nearest Neighbor in searching for the alternative design by using WEKA as a data mining tool and developed three classification models (Ashari et al., 2013). Their experiments showed that the Decision Tree is fastest and KNN is the slowest classification technique. The reason they mentioned is that, in the Decision Tree, there is no calculation involved. The classification by following the tree rules is faster than the ones that need calculation in the Naïve Bayes and KNN. Moreover, KNN is the slowest classifier because the classification time is directly related to the number of data. If the data size is bigger, larger distance calculation must be performed and this makes KNN extremely slow. They concluded that Naive Bayes outperformed Decision Tree and KNN in terms of accuracy, precision, recall and F-measure. Jebaseeli and Kirubakaran in 2012 investigated the use of three classifiers namely Naïve Bayes, KNN and random forest for prediction of opinions as positive or negative about the M learning system for the

purpose to analyze the efficiency of these three classifiers. A training data set containing 300 opinions was taken in the study with a split of 100 positive, 100 negative and 100 neutral opinions (Jebaseeli and Kirubakaran, 2012). In the preprocessing step, commonly occurring words and rarely occurring words were removed by using the SVD approach. SVD is used to rate the importance of words. The resultant preprocessed data were used as input for random forest algorithm. In this experiment a range of 55–60% accuracy was achieved. Khushboo et al. in 2012 used a counting based approach for opinion mining for English language. Total numbers of negative and positive words were used and then compared (Khushboo et al., 2012). In this study, naïve Bayesian algorithm was used and observed that if the dictionary is good then, it really gives good results. For increasing the accuracy of this algorithm, it is changed in the terms of parameters which are passed to the algorithm. Zhang et al. in 2008 worked on Chinese opinion mining by using machine learning approach (Zhang et al., 2008). Three classifiers SVM, Naïve Bayes Multinomial and Decision Tree were used to train the labeled corpus to learn certain classification functions. For this purpose, AmazonCN review dataset was used. It was found that the performance of the proposed system was satisfied while using SVM with String Kernel. Abbasi et al. in 2008 suggested sentiment analysis methodologies for the classification of opinions posted on the web forum in Arabic and English languages (Abbasi et al., 2008). In this research, specific feature extraction components were used that were integrated to account for the linguistic characteristics of Arabic language. The proposed system was very good in accuracy (93.62%). However, the domain was very specific because this system only classified sentiments related to hate and extremist groups' forums, as hate and extremist vocabulary is limited and it is not difficult to distinguish between positive and negative words. Moreover, there was no preprocessing step involved which is very important for Arabic language. Pang et al. in 2002 classified documents by overall sentiment rather than topic to determine whether a review is positive or negative. Movie reviews were used as dataset. It was found that standard machine learning methods absolutely outperform human produced baseline (Pang et al., 2002). However, their results showed that Naïve Bayes, maximum entropy classification and SVM do not perform as well on sentiment classification as on traditional topic-based categorization. Syed et al. in 2014 developed a framework that was based on grammatical model. This approach focused on grammatical structure of sentences and morphological structure of the words. In this technique, two types of grammatical structures were extracted and linked, the adjective phrases and nominal phrases. Adjective phrases were termed as Senti-Units and nominal as their targets. Shallow parsing and dependency parsing methods were applied and found to have 82.5% accuracy (Syed et al., 2014).

All the above work on opinion mining is in the English language. Other than English, research has been carried out in Chinese, Arabic, Malay, and Japanese language. The literature suggests that less work has been done in Urdu language especially in the Roman version.

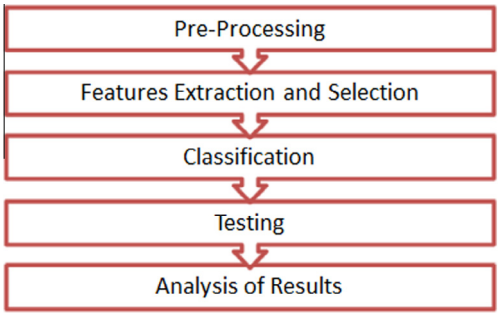

Figure 1 Proposed model.

3. Materials and methods

The proposed model is divided into five steps. First, opinions written in Roman-Urdu are extracted from a Blog using Easy Web Extractor software. The extracted opinions are documented in text files to prepare a training dataset containing 150 positive and 150 negative opinions, as labeled examples. The dataset is first converted into ARFF (Attribute-Relation File Format) by using Tex Directory loader command of WEKA in Simple CLI mode. The dataset in ARFF is then loaded to the WEKA explorer mode as training data set to train the machine. The data are first preprocessed using WEKA filters and then three different algorithms i.e. Naïve Bayesian, KNN and Decision Tree are applied on the dataset to train the machine and develop three models. Testing data set is supplied to the three models and the results in each case are analyzed. The following steps are followed in the methodology (Fig. 1).

3.1. Pre-processing

In the Pre-processing step, the data were prepared before it was being forwarded for classification to get accurate results. The following steps were used for preprocessing.

3.1.1. Extraction

The extraction process involves crawling in the specific web site for extracting information of interest. In this study, Easy Web Extractor is used to extract user comments posted on a Blog (http://hamariweb.com/blogs/blogdetails.aspx?id=59&Page=1). The blog contains public comments on "Effect of Facebook Usage". The users freely posted their comments mostly in multi-language, for example, "ye mobile **nice** hay", "ye cam achi **condition** me hay", "is mobile ke battery **life** ziada hay", etc. The reason is the influence of English Language in Urdu speaking community (Ahmed, 2009). Similarly, in this research different comments were posted on the topic in multiple languages. For example, "facebook aik **informative** website hay", "is website pe students apna sara time **waste** kartay hain", "is se taleb ilmo kee **study** par **negative** asar parta hy", "پڑھ نے والوںکےلئےف یسا یک ت دری سی وت فـری حی ذریعہہے" etc.

To start the process of extraction, first a project was created in Easy Web Extractor software and then following steps were followed:

Step 1: The URL of source website is entered in the input box and the web page is uploaded.
Step 2: The next button is pressed. It leads to the extraction pattern window where the area to be extracted is selected and HTML DOM is prepared for it.

Step 3: The first record (opinion) is selected as data-columns.
Step 4: Select next-page-urls to reach other pages.
Step 5: Click on extract button to extract the data from all pages
Step 6: Export extracted data to your computer and make the dataset.

3.1.2. Development of corpus

Each extracted opinion was stored in separate text file by using Easy Web Extractor software. It gave a set of text files. The text files were placed in two different folders such that, text files having positive opinions were placed in the positives folder and text files containing negative opinions were placed in the negatives folder. In this research, 150 positive and 150 negative comments were taken as training data set.

3.1.3. Conversion of extracted data into Arff

The WEKA software was used for Pre-processing, classification, building models of the training data set and predicting the polarity of the testing data set. The WEKA accepts data in Attribute-Relation File Format (ARFF). Therefore, the training set composed of 300 text files (150 positive and 150 negative) was converted into a single Attribute-Relation File Format (ARFF) by using the following Text Directory Loader command in Simple CLI mode of WEKA.

```
>java  weka.core.converters.TextDirectoryLoader–dir/C:/
opinions/trainingset >
C:/opinionmining/trainingdataset.arff
```

This command loads all text files in a directory and uses the subdirectory names as class labels. In our case, subdirectories were positive and negative, which reflected as class labels in ARFF. The content of the text files were stored in a String attribute tagged with relevant class labels.

3.2. Features extraction and selection

In the case of text classification, the features (attributes) are the terms (word tokens) which are large in number and affect the efficiency in terms of time taken to build the model. So feature reduction is necessary. It serves two purposes. First it decreases the size of effective vocabulary which makes it efficient to apply a classifier on the training data and second, it eliminates noise features which decrease classification errors on new data.

In this research study, both feature selection and extraction were performed by using WEKA filters under the preprocess tab.

3.2.1. StringToWordVector filter

StringToWordVector filter is used to transform String attributes into a set of attributes that represent word occurrence depending on the tokenizer used. In this research, StringToWordVector filter was used to transform the text (loaded to the weka through TextDirectory loader command) into a set of word tokens by setting certain parameters discussed below.

3.2.1.1. TF–IDF transform. TF–IDF stands for Term Frequency–Inverse Document Frequency. It is used to assign weights to the terms which have relative importance in a corpus (Rajaraman and Ullman, 2011). TF–IDF value increases as the occurrence of a word appears frequently in a document but is offset by the repetition of a word in the corpus, which helps to adjust for the fact that some words appear more frequently in general. In this research, the irrelevant terms were eliminated using TF–IDF:

$$\mathrm{tf}(t,d) = 0.5 + \frac{0.5 \times \mathrm{f}(t,d)}{\max\{\mathrm{f}(w,d) : w \in d\}} \tag{1}$$

Some terms like "the" are so common, which occurrs in almost each document. According to Term Frequency (TF), the documents which use the term "the" more frequently will incorrectly get more weight without giving enough weight to more meaningful but less common terms like "Good", "Excellent" and "Bad". Therefore, an Inverse Document Frequency (IDF) factor was combined with TF to moderate the weight of terms that occur frequently in the document set and to increase the weight of terms that occur rarely:

$$\mathrm{idf}(t,D) = \log \frac{N}{|\{d \in D : t \in d\}|} \tag{2}$$

here:

N represents total no. of documents in the corpus.
$|\{d \in D : t \in d\}|$: represents no. of documents where the term t occurs (i.e., $\mathrm{tf}(t,d) \neq 0$).

3.2.1.2. Term Frequency–Inverse Document Frequency. Then TF–IDF is calculated as

$$\mathrm{tfidf}(t,d,D) = \mathrm{tf}(t,d) \times \mathrm{idf}(t,D) \tag{3}$$

3.2.1.3. Lower case tokens. Lower case token parameter is used to convert all word tokens to lower case before adding to the dictionary. The purpose of setting this parameter is to shift all words to a single format that can easily be used for prediction.

3.2.1.4. Minimum term frequency. This parameter allows the users to specify a minimum value of occurrence of a word token for its consideration in feature selection. If we set the value 2 in minimum term frequency then the word tokens that occur less than 2 times will not be considered in features selection. In our case, occurrence of a value of at least one was important therefore, the value was set as 1.

3.2.1.5. Output Words Count. This parameter is used to count the number of occurrences of a word token in a single document. For example, a word comes three times in a single document that in vector matrix will reflect the value. In this research, Output Words Count was not used because the features were converted into binarized form which indicates that if a word appears in the opinion its value is 1 otherwise 0.

3.2.1.6. Tokenizer. A simple tokenizer that is using the java. util.StringTokenizer class to break the strings into word tokens. The Tokenizer create tokens out of string by reading delimiters \r\t\n.,;:\'\"()?!-> <#$\\%&*+/@^_=[]{}|'~01 23456789. The following is the code of tokenizer.

```
//Make a filter
StringToWordVector filter = new StringToWordVector();
//Make a tokenizer
WordTokenizer wt = new WordTokenizer();
String delimiters = "\r\t\n.,;:\'\"()?!-> <#$\\%&*+/@
^_=[]{}|`~0123456789";
wt.setDelimiters(delimiters);
filter.setTokenizer(wt);
//Inform filter about dataset
filter.setInputFormat(data);
```

3.2.1.7. WordsToKeep. This option enables us to restrict a specific number of words per class. In general, it is good for the classifier to keep as many words as possible with small frequencies. But keeping too many words as features is badly affecting the efficiency of classifiers, because large numbers of features (attributes) make the classifiers to take a longer time in building model. However, different filters like TF–IDF keep most predictive words.

3.2.2. Reorder filter

After applying the StringToVector filter, the string attribute was converted into word tokens each with a specific value counted as TF–IDF value. Class attribute remained the first token in the list. As WEKA considers the last attribute as a class attribute. Hence, Reorder filter was used to relocate the class attribute to the end where the WEKA reads it as a class attribute. The following command was used:

weka.filters.unsupervised.attribute.Reorder: Reorder-R 2-last,1

Reorder filter generates the output with a new order of the attributes. It is useful if one wants to move an attribute to the end to use it as class attribute (e.g. with using "-R 2-last,1").

3.2.3. Numeric to binary filter

Numeric attributes were converted into binary attributes by using NumericToBinary filter of WEKA. This filter converted all numeric attributes into binary attributes (apart from the class attribute). If the value of the numeric attribute is exactly zero, the value of the new attribute will be zero otherwise it will be 1. Its syntax is:

weka.filters.unsupervised.attribute.NumericToBinary

The presence of a word token in an opinion document is represented by 1 and its absence in the document is represented by 0.

3.2.4. Bag of Words Model

In Bag of Words Model, a document is represented as an unordered collection of words, regardless of the grammar and even word order. For natural language processing, a document is represented as a Bag (multiset) of its words regardless of grammar and word order but keeping multiplicity. Bag of Words Model is used commonly in methods of document classification and frequency of occurrence of each word is used as a feature for the training classifier.

Consider two text documents:

Ali likes to use facebook. Maryam likes facebook too.
Ali also likes to watch movie.

On the basis of above two texts documents, a dictionary is constructed as:

```
{
   "Ali": 1,
   "likes": 2,
   "to": 3,
   "uSe": 4,
   "facebook": 5,
   "also": 6,
   "movie": 7,
   "watch": 8,
   "Maryam": 9,
   "too": 10
}
```

It has ten distinct words. By using the index of the dictionary, each document is represented by a 10 entry vector:

Vector-1: [1,2,1,1,2,0,0,0,1,1]
Vector-2: [1,1,1,0,0,0,1,1,0,0]

Here each entry of the vector refers to the number of occurrences of the corresponding word in the dictionary. For example, vector-1 represents the first document and its first and second entries are "1, 2". The first entry corresponds to the word "Ali", which is the first word in the dictionary having value "1", which shows that "Ali" appears in the first document one time. Similarly, the second entry refer to the word "likes" which is the second word in the dictionary having value "2", which shows that "likes" appears in the first document two times, however this vector representation does not follow the order of the words in original sentences.

In the binarized form the presence of a word token is represented by 1 and its absence is represented by 0. By this way, the above vectors are written as (Fig. 2):

Vector-1: [1,1,1,1,1,0,0,0,1,1]
Vector-2: [1,1,1,0,0,0,1,1,0,0]

3.3. Classification

Classification is a technique of assigning class label to a set of unclassified cases (Shrivastava, 2014). There are two types of classification:

(i) Supervised classification.
(ii) Unsupervised classification.

(i) Supervised classification: In supervised classification, class labels are known in advance. Training data is a set of records having multiple attributes including the class attribute that has predefined class labels. In this technique, a model is developed by analyzing the training dataset. The model is used to assign class labels to the testing dataset.

(ii) Unsupervised classification: In this type of classification, class labels are not known in advance. After classification,

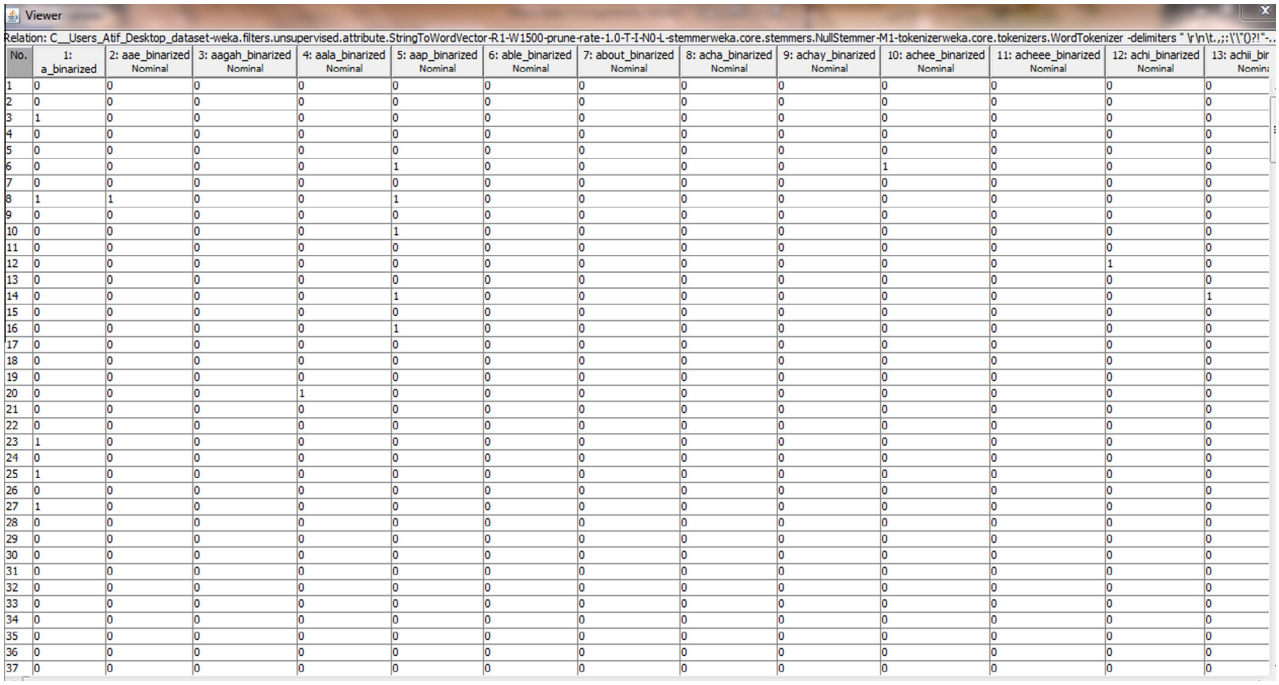

Figure 2 Vector matrix.

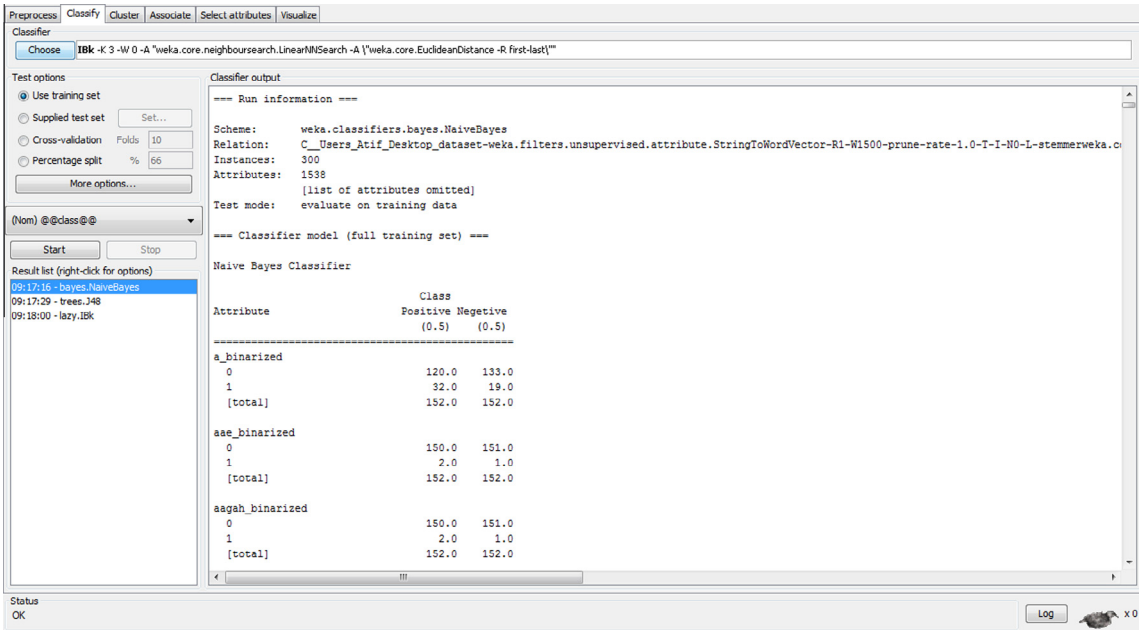

Figure 3 Building models in WEKA.

records are assigned class labels by grouping records on the basis of some natural similarities. Clustering is unsupervised classification.

3.3.1. Building classification models of Naïve Bayesian, J48 and KNN

For this purpose, "Classify tab" of the WEKA tool was used. There are 53 classification algorithms available in "Classify tab" of WEKA version 3.7.10. The machine was trained using a training data set by using three classification techniques e.g. Naïve Bayesian, Decision Tree (J48 in WEKA) and *k*-Nearest Neighbor (IBk in WEKA) and three classification models were built. These models were supplied with testing data to predict polarity of opinions, as positive or negative (Fig. 3).

3.4. Testing of models

The models were supplied using testing dataset. The test data set is the collection of new opinions posted on the blog. The

Table 1 Accuracy of Naïve Bayesian on training set.

	Number	Age (%)
Correctly classified instances	292	97.33
Incorrectly classified instances	8	2.67
Total number of instances	300	100

Table 2 Contingency table of Naïve Bayesian on training set.

	X	Y
A	149	1
B	7	143
149 (TP)		1 (FN)
7 (FP)		143(TN)

Table 3 Summarized results of Naïve Bayesian on training set.

Class	TP rate	FP rate	Precision	Recall	F-measure	ROC area
Positive	0.993	0.047	0.955	0.993	0.974	0.999
Negative	0.953	0.007	0.993	0.953	0.973	0.999
Overall (aggregate)	0.973	0.027	0.974	0.973	0.973	0.999

testing data set was also preprocessed and converted to ARFF file. The test dataset was loaded to the WEKA by using "Classify tab" and selected "Supplied Test Set" option in "Test Options" panel. After that the below steps were performed:

Step 1: Click on "Set..." button.
Step 2: Click on "Open File".
Step 3: Selected test dataset file.
Step 4: Click on "Close" button.
Step 5: Right click on each model and select option "Re-evaluate model on current test set".

3.5. Analysis of results

In this step, the results were analyzed to find out how accurately the classification model classified the incoming opinions. All three classification models were run on same testing data and the results were compared to determine which classifier more accurately classified the testing data. The results are analyzed and evaluated by using standard methodologies of information retrieval i.e. precision, recall and F-measure.

4. Results and discussion

This section describes the results of experiments carried out in this research along with discussions on these outcomes in light of set objectives outlined in the study. Three classification algorithms are used to check the performance of three algorithms. The algorithm which gives better accuracy, recall and F-measure value is considered the most efficient when applied to the training data set as well as testing data set size.

4.1. Classification by Naïve Bayesian

The training data set comprised of 150 positive and 150 negative opinion's documents were taken to build a classification model based on Naïve Bayesian algorithm. The data was pre-processed using weka built-in filters as discussed in the previous section. The pre-processed form of training data was

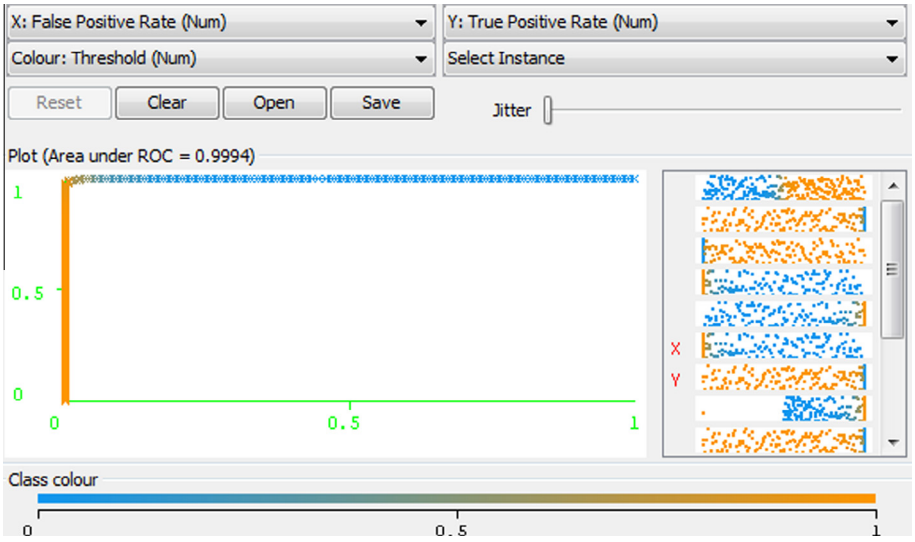

Figure 4 ROC curve.

Table 4 Accuracy of Naïve Bayesian on testing set.

	Number	Age (%)
Correctly classified instances	39	97.5
Incorrectly classified instances	1	2.5
Total number of instances	40	100

Table 5 Contingency table of Naïve Bayesian on testing set.

	X	Y
A	20	0
B	1	19

Table 6 Summarized results of Naïve Bayesian on testing set.

Class	TP rate	FP rate	Precision	Recall	F-measure	ROC AREA
Positive	1.000	0.050	0.952	1.000	0.976	1.000
Negative	0.950	0.000	1.000	0.950	0.974	1.000
Overall (aggregate)	0.975	0.025	0.976	0.973	0.975	1.000

uploaded in WEKA Explorer interface. The Classify tab enabled us to choose Naïve Bayes classifier.

4.1.1. Building model on training dataset

After choosing Naïve Bayes classification algorithm in weka classify tab, the algorithm was applied on the training set to build the model. The results are shown in Tables 1–3 and Fig. 4.

4.1.2. Testing model on testing dataset

After building the classification model on training set using Naïve Bayes algorithm, the testing data set was supplied to the model and performed testing by using weka option "Re-evaluate model on current test set". Results are shown in Tables 4–6 and Fig. 5.

4.2. Classification by Decision Tree

The training data set comprising 150 positive and 150 negative opinion's documents was again taken to build a classification model based on Decision Tree. The pre-processed form of training data was uploaded in WEKA Explorer interface. The Classify tab enabled us to choose the J48 algorithm, which is used for Decision Tree.

4.2.1. Building model on training dataset

After choosing J48 algorithm in weka, the algorithm was applied on a training set to build the model. Results are shown in Tables 7–9 and Fig. 6.

Table 7 Accuracy of Decision Tree on training set.

	Number	Age (%)
Correctly classified instances	284	94.667
Incorrectly Classified Instances	16	5.333
Total number of instances	300	100

Table 8 Contingency table of Decision Tree on training set.

	X	Y
A	140	10
B	6	144

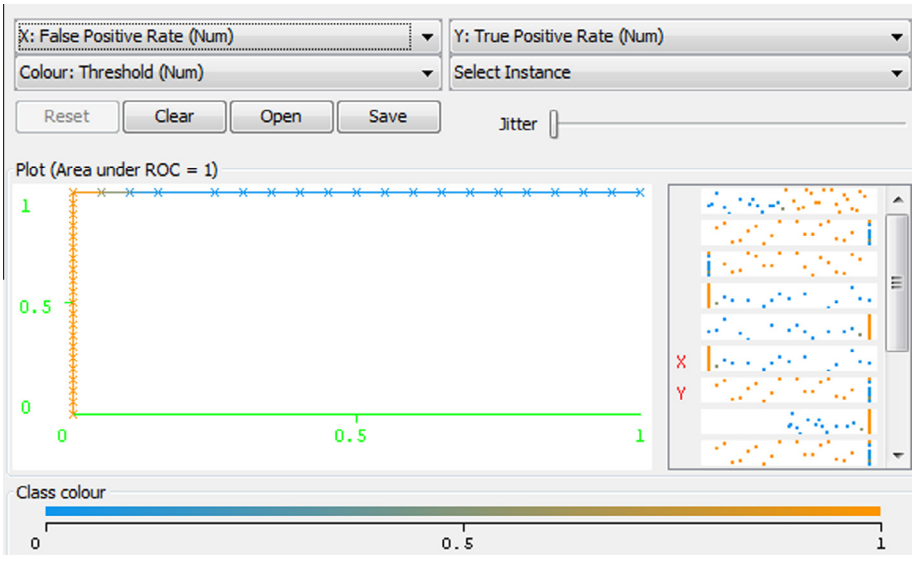

Figure 5 ROC curve.

Table 9 Summarized results of Decision Tree on training set.

Class	TP rate	FP rate	Precision	Recall	F-measure	ROC area
Positive	0.933	0.040	0.959	0.933	0.946	0.969
Negative	0.960	0.067	0.935	0.960	0.947	0.969
Overall (aggregate)	0.947	0.053	0.947	0.947	0.947	0.969

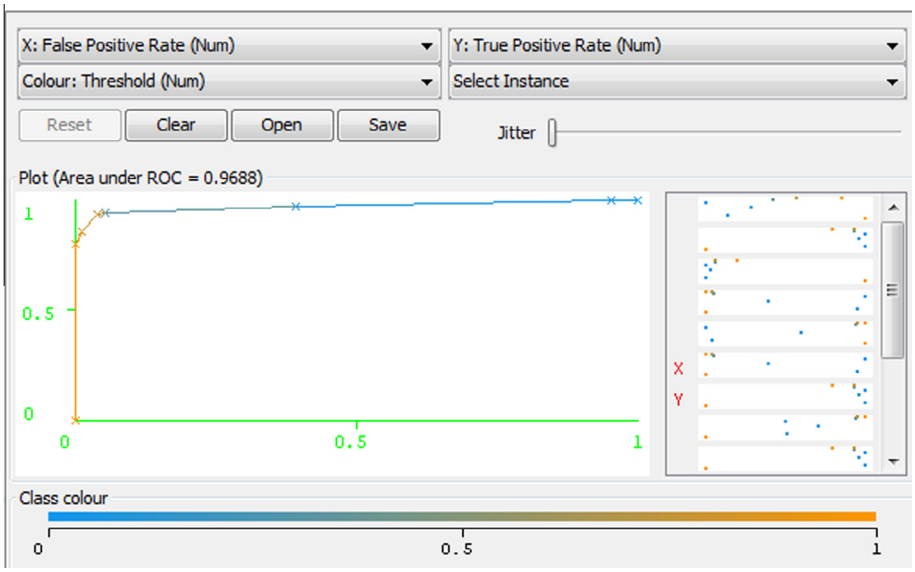

Figure 6 ROC curve.

Table 10 Accuracy of Decision Tree on testing set.

	Number	Age (%)
Correctly classified instances	37	92.5
Incorrectly classified instances	3	7.5
Total number of instances	40	100

Table 11 Contingency table of Decision Tree on testing set.

	X	Y
A	18	2
B	1	19

4.2.2. Testing model on testing dataset

After building the classification model on a training set using Decision Tree algorithm (J48), the testing data set was supplied to the model and performed testing by using weka option ''Re-evaluate model on current test set''. Results are shown in Tables 10–12 and Fig. 7.

4.3. Classification by KNN

The training data set comprising of 150 Positive and 150 Negative opinion's documents was taken again for the third algorithm to build a classification model based on KNN. The pre-processed form of training data was uploaded in WEKA Explorer interface. The Classify tab enabled us to choose the IBk algorithm, which is used for implementing the KNN classification.

4.3.1. Building model on training dataset

After choosing IBk algorithm in weka, and setting essential parameters (i.e. $k = 3$, search algorithm used = linear NN

Table 12 Detailed results of Decision Tree on testing set.

Class	TP rate	FP rate	Precision	Recall	F-measure	ROC area
Positive	0.900	0.050	0.947	0.900	0.923	0.948
Negative	0.950	0.100	0.905	0.950	0.927	0.948
Overall (aggregate)	0.925	0.075	0.926	0.925	0.925	0.948

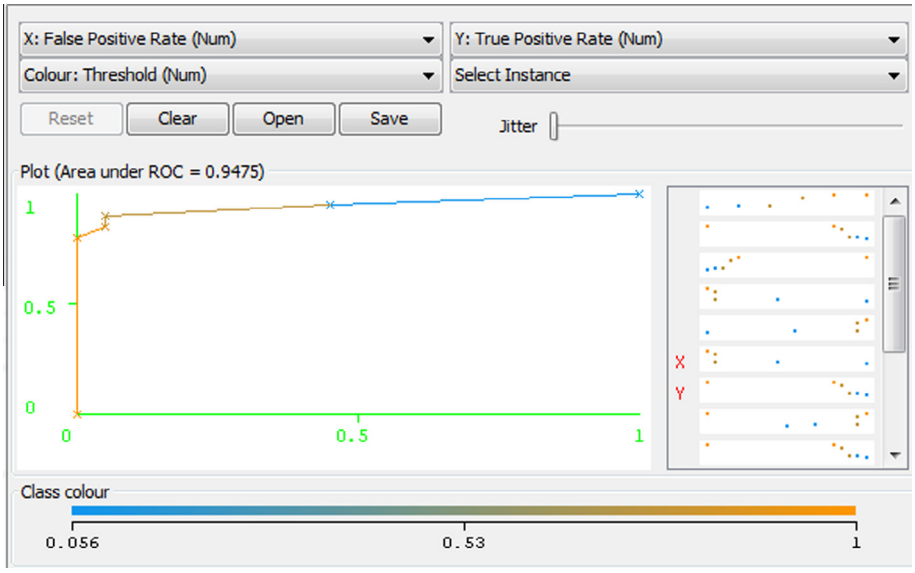

Figure 7 ROC curve.

Table 13 Accuracy of KNN on the training set.

	Number	Age (%)
Correctly classified instances	260	86.667
Incorrectly classified instances	40	13.333
Total number of instances	300	100

Table 14 Contingency table of KNN on the training set.

	X	Y
A	134	16
B	24	126

search, distance function = Euclidean distance), the algorithm was applied on a training set to build the model. Results are shown in Tables 13–15 and Fig. 8.

4.3.2. Testing model on testing dataset

After building the classification model on a training set using KNN Algorithm (IBk), the testing data set was supplied to the model and performed testing by using weka option "Re-evaluate model on current test set". Results are shown in Tables 16–18 and Fig. 9.

4.4. Comparison of results

After applying three algorithms on the same data set, the following results were obtained. These results are combined in a table for performance comparison of these algorithms.

From Table 19, the following results are obtained:

(1) Naïve Bayes algorithm performed best in classification of Roman Urdu opinions in terms of higher accuracy, higher precision, higher recall and higher value of F-measure as compared to the Decision Tree and KNN.
(2) The precision of KNN decreases significantly as the sample size increases, however, values of recall and F-measure initially increase and then decrease gradually with an increase in sample size.

Table 15 Summarized results of KNN on the training set.

Class	TP rate	FP rate	Precision	Recall	F-measure	ROC area
Positive	0.893	0.160	0.848	0.893	0.870	0.936
Negative	0.840	0.107	0.887	0.840	0.863	0.936
Overall (aggregate)	0.867	0.133	0.868	0.867	0.867	0.936

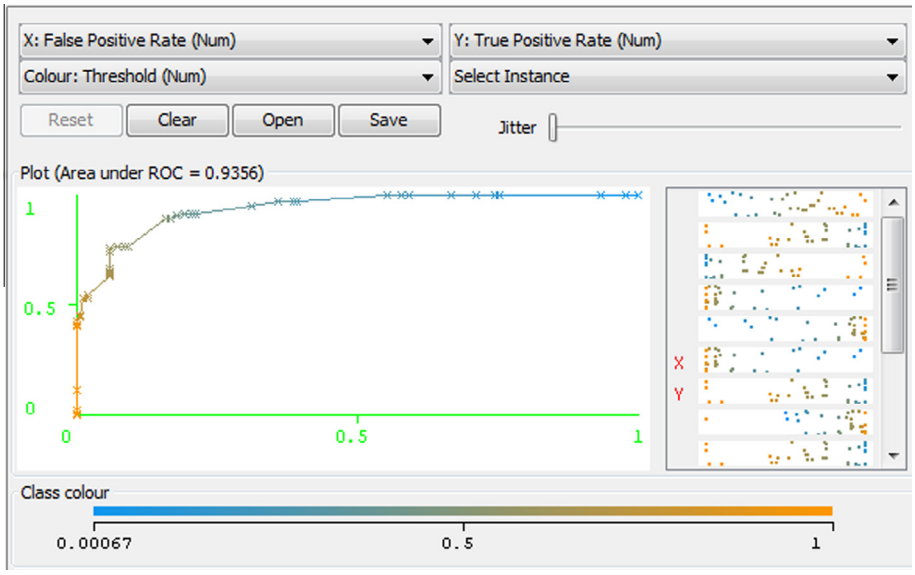

Figure 8 ROC curve.

Table 16 Accuracy of KNN on the testing set.

	Number	Age (%)
Correctly classified instances	38	95
Incorrectly classified instances	2	5
Total number of instances	40	100

Table 17 Contingency table of KNN on the testing set.

	X	Y
A	19	1
B	1	19

(3) The recall of Decision Tree increases significantly as the sample size increases, however, values of precision and F-measure initially increase and then decrease gradually with an increase in sample size.

These results expressed in the form of graphs are shown in Figs. 10–15.

4.5. Discussion

From the above experiments, it is revealed that Naïve Bayes out performs the rest of the two algorithms i.e. Decision Tree and KNN. Its performance is best in terms of accuracy, precision, recall and F-measure.

Decision Tree is the fastest and KNN is the slowest classification technique (Ashari et al., 2013). The reason is that, there is no calculation process involved in Decision Tree. It performs classification by following certain tree rules which are faster than the calculation involved in Naïve Bayes and KNN. On the other hand, KNN is the slowest of the three mentioned classifiers because its classification time is directly related to the size of data. It means if the the data size is bigger, a larger distance calculation would be performed and this is what makes KNN extremely slow.

Naïve Bayes is a simple classifier but it can perform much better than other sophisticated classification algorithms. It is fast and accurate, even applied to large datasets (Han and Kamber, 2001). It has good speed during learning and predicting. Its learning time is linear to the number of examples and its prediction time is independent of the number of examples (Pazzani and Bilsus, 1997). As far as computation is concerned, Naïve Bayes is more efficient in learning and classification than Decision Tree (Amor et al., 2004). The reason behind this fact is that it shows a good probability estimate for correct class, which enables it to perform the correct classification (Domingos and Pazzani, 1996). Another reason for good performance of Naïve Bayes over the other two classification techniques is 0–1 loss function that defines the error as the number of incorrect predictions in Naïve Bayes. Unlike other loss functions, it does not penalize for inaccurate classification as long as the greatest probability is assigned to the correct class.

Table 18 Summarized results of KNN on the testing set.

Class	TP rate	FP rate	Precision	Recall	F-measure	ROC area
Positive	0.950	0.050	0.950	0.950	0.950	0.976
Negative	0.950	0.050	0.950	0.950	0.950	0.976
Overall (aggregate)	0.950	0.050	0.950	0.950	0.950	0.976

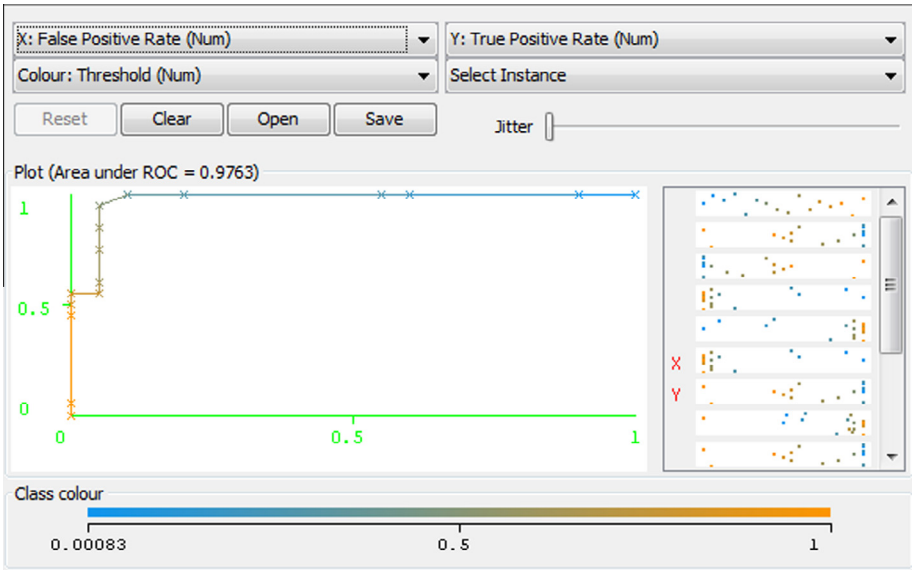

Figure 9 ROC curve.

Table 19 Comparison of results.

Algorithms	Data set	Time taken (in s)	Accuracy (%)	Precision	Recall	F-measure	ROC area
Naïve Bayesian	Training	0.09	97.33	0.974	0.973	0.973	0.999
	Testing	0.01	97.50	0.976	0.973	0.975	1.000
Decision tree	Training	0.02	94.67	0.947	0.947	0.947	0.969
	Testing	0.00	92.50	0.926	0.925	0.925	0.948
KNN	Training	0.13	86.67	0.868	0.867	0.867	0.936
	Testing	0.04	95.00	0.950	0.950	0.950	0.976

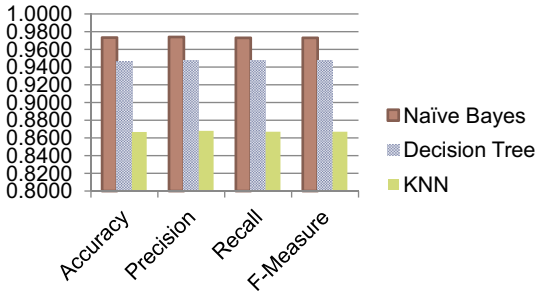

Figure 10 Comparison of results of three algorithms on training dataset.

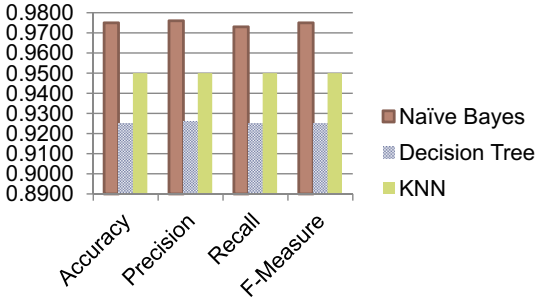

Figure 11 Comparison of results of three algorithms on testing dataset.

Another reason for Naïve Bayesian's good performance is that in a data set two attributes may depend on each other but this dependence may distribute equally in each class. Similarly, when dependencies among all attributes work together they may cancel out the effect of each other and hence dependencies no longer affect the classification. This is the reason because of which conditional independence assumption is violated.

The experiment shows that k-Nearest Neighbor is worse than both Naïve Bayes and Decision Tree. The KNN uses the number of nearest neighbor "k" as one of the parameters in classifying an object. The value of k along with distance function and weighting function affects the performance of the classifier (Batista and Silva, 2009). In this study, we took $k = 3$. When greater values of k were tested i.e. $k = 5, 7, 9,$

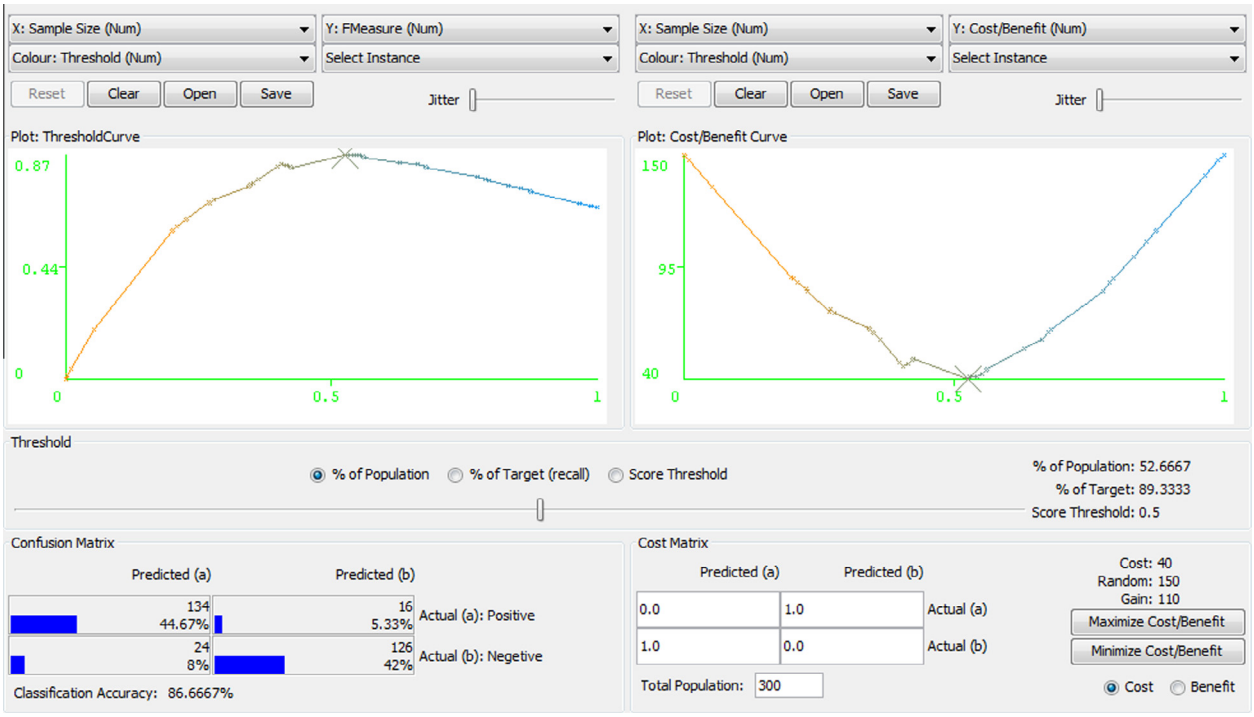

Figure 12 Effect of sample size on F-measure in KNN.

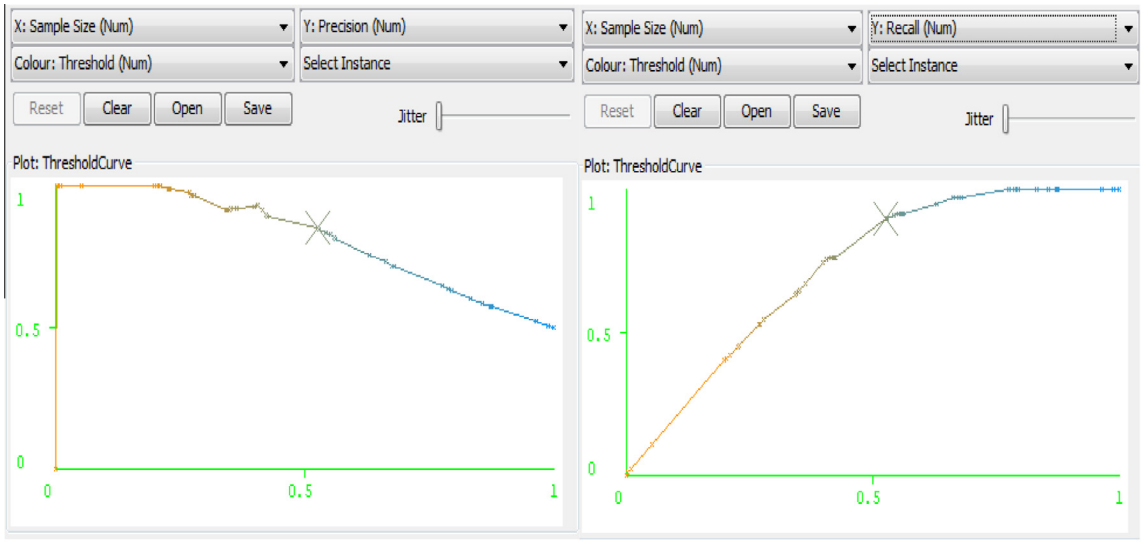

Figure 13 Effect of sample size on precision and recall in KNN.

the performance of the classifier moved downward. For all weighting functions and distance function, the performance of KNN decreases for higher values of k. One weakness of KNN is its slow runtime and large memory requirement (Bay, 1999) because the k-NN classifier requires a large memory to store the entire training set (Lee, 1991). It means the larger the data set, the more memory it will require to store the training data and subsequently, larger distance calculations would be performed, which makes it slow and inefficient than others.

The recall of the Decision Tree increases significantly as the sample size increases, however, values of precision and F-measure initially increase and then decrease gradually with an increase in sample size. It means that the larger the training set the larger will be the tree size and more accurate results will be obtained than the tree built from subsets (Catlett, 1991). The reason for its increased accuracy and recall is because of extra size of the tree and training instances. This provides extra rules and allows better choices of attribute while building the tree and better choices of the sub trees to prune after it has been built.

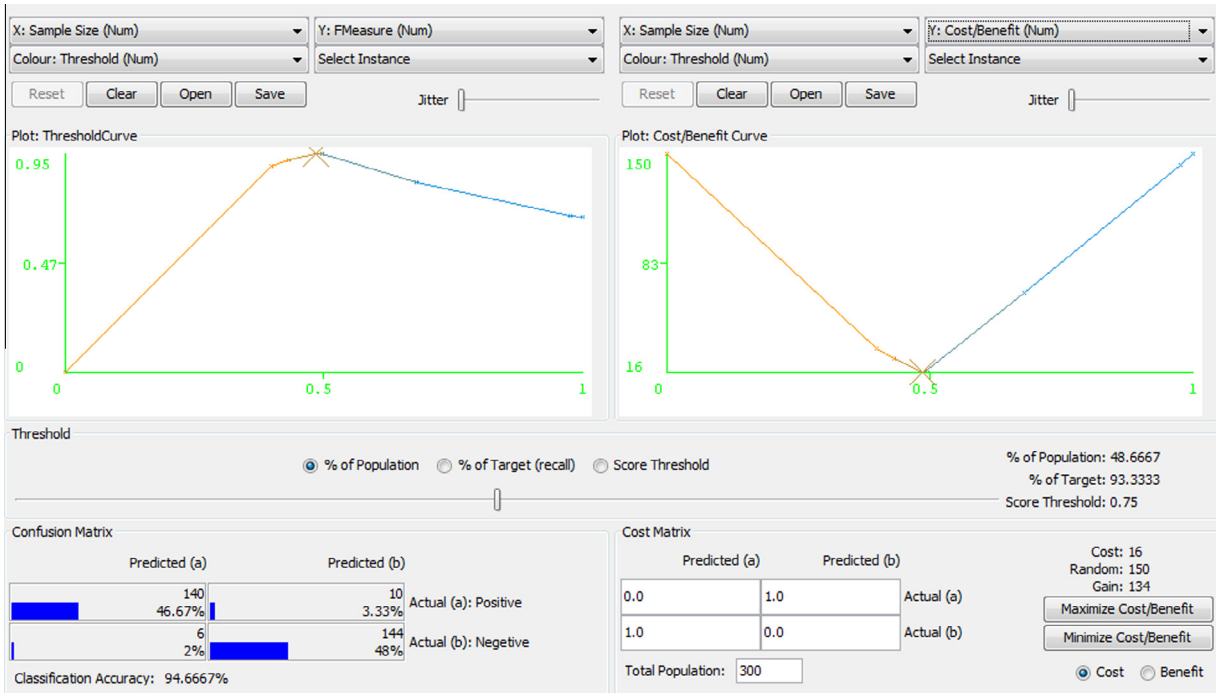

Figure 14 Effect of sample size on F-measure in Decision Tree.

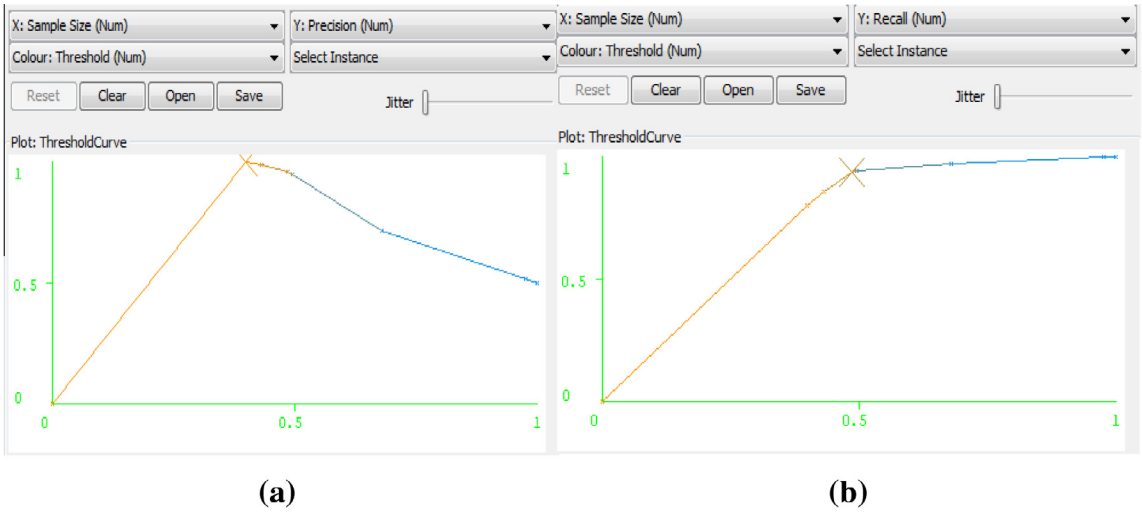

Figure 15 Effect of sample size on precision and recall in Decision Tree.

5. Conclusion

This research was conducted on Roman Urdu opinion mining by using three classification algorithms i.e. Naïve Bayes, Decision Tree and KNN. A training dataset was used containing 150 positive and 150 negative opinions as labeled examples, to train the machine and to develop three models. Testing dataset was supplied to three different models for classification. The results show that Naïve Bayes algorithm performed best in terms of higher accuracy, higher precision, higher recall and higher value of F-measure as compared to the Decision Tree and KNN.

As far as computation is concerned, Naïve Bayes is more efficient in learning and classification than the Decision Tree (Amor et al., 2004). The reason for its good performance is that, in many cases the probability estimates may be poor, but the correct class will still have highest estimate, leading to correct classification (Domingos and Pazzani, 1996). Another reason for good performance of Naïve Bayesian over other two is zero-one loss function used in Naïve Bayesian classification.

Decision Tree is the fastest and KNN is the slowest classification technique (Ashari et al., 2013). The reason is that, there is no calculation process involved in Decision Tree. The precision of KNN decreases significantly as the sample

size increases. However, values of recall and F-measure initially increase and then decrease gradually with an increase in sample size. As KNN uses number of nearest neighbor "*k*" as one of the parameters, in classifying an object and this value of *k* affects the performance of the classifier (Batista and Silva, 2009). The recall of Decision Tree increases significantly as the sample size increases, however, values of precision and F-measure initially increase and then decrease gradually with an increase in sample size. This means that trees built from very large training sets are larger and more accurate than trees built from subsets (Catlett, 1991).

References

Abbasi, A., Chen, H., Salem, A., 2008. Sentiment analysis in multiple languages: feature selection for opinion classification in Web forums. ACM Trans. Inf. Syst. 26 (3). http://dx.doi.org/10.1145/1361684.1361685.

Ahmed, T., 2009. Roman to Urdu transliteration using wordlist. In: Proceedings of the Conference on Language and Technology, 305–309.

Amor, N.B., Benferhat, S., Elouedi, Z., 2004. Naive Bayes vs decision trees in intrusion detection systems. In: ACM Symp. on Applied Computing, pp. 420–424.

Ashari, A., Paryudi, I., Tjao, A.M., 2013. Performance comparison between Naïve Bayes, decision tree and *k*-nearest neighbor in searching alternative design in an energy simulation tool. Int. J. Adv. Comput. Sci. Appl. 4 (11), 33–39.

Batista, G.E.A.P.A., Silva, D.F., 2009. How k-nearest neighbor parameters affect its performance. In: Simposio Argentino de Inteligencia Artificial, pp. 95–106.

Bay, S.D., 1999. Nearest neighbor classification from multiple feature subsets. Intell. Data Anal. 3 (3), 191–209.

Catlett, J., 1991. Overpruning large decision trees. In: Proceedings of Int. Joint Conf. on Artif. Intel. (IJCAI).

Daud, M., Khan, R., Duad, A., 2014. Roman Urdu opinion mining system (RUOMiS). CSEIJ 4 (6), 1–9. http://dx.doi.org/10.5121/cseij.2014.4601.

Domingos, P., Pazzani, M., 1996. Beyond independence: conditions for the optimality of the simple Bayesian classifier. In: 13th International Conference on Machine Learning, 105–112.

Han, J., Kamber, M., 2001. Data Mining: Concepts and Techniques. Morgan–Kaufmann Publishers, San Francisco, 310–315.

Jebaseeli, A.N., Kirubakaran, E., 2012. M-learning sentiment analysis with data mining techniques. Int. J. Comput. Sci. Telecommun. 3 (8), 45–48.

Kaur, A., Gupta, V., 2014. N-gram based approach for opinion mining of Punjabi text, multi-disciplinary trends in artificial intelligence. Lecture Notes Comput. Sci. 8875, 81–88.

Khushboo, T., Vekariya, S.K., Mishra, S., 2012. Mining of sentence level opinion using supervised term weighted approach of Naïve Bayesian algorithm. Int. J. Comput. Technol. Appl. 3 (3), 987–991.

Lee, Y., 1991. Handwritten digit recognition using *k* Nearest-Neighbor, radial-basis function, and back propagation neural networks. Neural Comput. 3 (3), 440–449.

Pang, B., Lee, L., 2008. Opinion mining and sentiment analysis. Found. Trends Inf. Retrieval 2 (1–2), 1–135.

Pang, B., Lee, L., Vaithyanathan, S., 2002. Sentiment classification using machine learning techniques. In: Proceedings of the ACL-02 Conference on Empirical Methods in Natural Language Processing, vol. 10, pp. 79–86.

Pazzani, M., Bilsus, D., 1997. Learning and revising user profiles: the identification of interesting web sites. Mach. Learn. 27, 313–331.

Rajaraman, A., Ullman, J.D., 2011. Data mining. Mining of Massive Datasets. pp. 1–17.

Rashid, A., Anwer, N., Iqbal, M., Muhammad, S., 2013. A survey paper: areas, techniques and challenges of opinion mining. Int. J. Comput. Sci. 10 (6), 18–31.

Shrivastava, P., 2014. A study on some of data warehouses and data mining (case study of data mining for environmental problems). Int. J. Comput. Sci. Trends Technol. (IJCST) 2 (1), 36–43.

Syed, A.Z., Aslam, M., Martinez-Enriquez, A.M., 2014. Associating targets with SentiUnits: a step forward in sentiment analysis of Urdu text. Artif. Intell. Rev. 41 (4), 535–561.

Zhang, C., Zuo, W., Peng, T., He, F., 2008. Sentiment classification for Chinese reviews using machine learning methods based on string kernel. In: Third 2008 International Conference on Convergence and Hybrid Information Technology, pp. 909–914.