IOSR Journal of Computer Engineering (IOSRJCE)
ISSN: 2278-0661, ISBN: 2278-8727 Volume 5, Issue 6 (Sep-Oct. 2012), PP 30-36
www.iosrjournals.org

Different Similarity Measures for Text Classification Using Knn

[1]P.Sowmya Lakshmi, [2]V.Sushma, [3]T.Manasa
1, 2, 3(M Tech(Software Engineering) Dept .of Computer Science & Technology,(Student) Sreenidhi Institute of Science & Technology/An Autonomous Institution ,Hyderabad,AP, India)

***Abstract*:** *Present days humans are associated with many electronic gadgets which generate large amount of data on regular basis. The sole purpose of generated data was to meet the immediate needs and no attempt in organizing the data for later efficient retrieval was attempted. Over the period of time, the data generated became voluminous, this paper attempts to classify the huge data into different categories for easy retrieval. We have many techniques to classify the data which exists in the structured format, but not much work has been addressed when the data is available in textual form. In the present paper an attempt to classify the textual data based on its content is explored. The paper explores the process of building multi-classifier model for textual data. In the process of designing the model the K-Nearest Neighbor paradigm was employed, which has given encouraging results. The paper also attempts to explore different similarity measures, different feature selection techniques in the process of designing textual multi-classification.*

Keywords*-unstructured data classification, similarity measures, Euclidean, Manhattan, Squared-Euclidean, chessboard, bray-Curtis*

I. Introduction

Today humans are associated with electronic gadgets in one way or the other, to make life comfortable. With the immediate need of the data generated from the gadgets, no thought of organizing the data for the later retrievals was attempted. Manually organizing the voluminous data is a very difficult task due to the following reasons. It is time consuming and boring for the humans such a repetitive task. Moreover human experts in the concerned domains are needed for such a task, two experts opinion could differ. So we need an automatic mechanism to address the problem of classifying the data into different categories. Reference [5] gives two techniques in the process of data classification. Supervised learning where one attempts to construct a model by using the training data which has the independent attributes along with the decision attribute. The model tries to learn how the decision attribute is dependent on the independent attributes. Once the model is designed the classifier is subjected to the unseen data, if the test data were classified properly with a reasonable threshold error, then the classifier is deemed to be constructed and learning is said to be achieved.

On the other hand unsupervised learning refers to achieving the learning where no training data is available; the given data is distributed into clusters based on the similarities in the data. In the process of building clusters, care is taken where the inter-cluster similarities are almost zero and intra-cluster similarities are maximized. When such a property is achieved in all the clusters then, the data in the same cluster are treated to belong to the same class. Among many applications of such a classification the simplest application with binary textual classification is to classify the mails into spam and hams (legitimate mails). The binary classifier could be extended, considering the case where the documents (mails) are to be classified based on its contents into multi-class (say mails to be forwarded to the respective sections from a single mail-id).

This paper is organized as follows. In section-2 lexicon generation process is discussed. Next in section-3 detailed explanation of 9 different types of vector generation techniques is discussed. In section-4 different similarity measures are discussed. In section-5 experimental implementation details are provided along with results, finally at the end of the paper in section-6 conclusions and future work are provided.

II. Lexicon Set

Given are a set of documents Dtrain and their class label set C the documents belong to. In the case of training data we are provided the information of each individual element of Dtrain belonging to which element of C. Using this information, test data which contains document Dtest but no corresponding class information, a classifier is to be built. Reference [2] gives the entire process of generating lexicon, which contains all the words available in the Dtrain.

Throughout the experiment it is assumed that the training data is noise-free, so no effort was put to remove the noise while performing the experiment, though we have found noise in the corpus. Reuters-21578 from [3] was used in the experiment, which had 5485 documents as training data split across 8 classes and 2189 documents for testing.

Lexicon set is a set which contains all the words in the document set. In order to reduce the no: of words in the lexicon set, words which are not useful from the classification perspective termed as stop words were not included. In order to reduce the words further, words root form only are considered as members of the lexicon set, so stemming is employed as in [9]. In this experiment the lexicon size obtained was 14,822, resulting in 14,822 unique words. In this experiment all words were considered, rather than removing some words as stated in [1].

III. Vectors Generation

This section deals with the process of generating different vectors, their motivations and techniques are also provided. In the process of textual classification, text data is converted to, rather mapped to numerical data. Different techniques of such mapping are employed; in the process of doing this the order of the words in the document is not considered as described in [6].

A. Binary Vector (Vector-1)

The motivation for binary vector is the presence of related terms ascertains the class the document belongs to. So in binary vectors the words appearance is considered as vital information, but the no: of times it appeared is treated unimportant. So the no: of elements in the vector is equal to the size of the lexicon which is 14,822. No: of vectors equal to the no: of documents. Using the above rule binary vector generation has 5485 rows (for 5485 documents) with 14,822 (lexicon size) columns was generated.

B. Frequency Vector (Vector-2)

Not just the presence and absence of the words, but frequency of words also play role in the decision of the class the document belongs to, this motivated for frequency vectors. So in frequency vector generation not just the presence is considered but also the no: of times it appeared is considered to be important. Using this rule, frequency vectors are generated with 5485 rows and 14,822 columns.

C. Normalization

In the process of textual classification an attribute, which is a word should not play complete dominating role due to its large magnitude when compared with other attributes. In
order to consider this we need to normalize the values by scaling the entries bounded to fall within a small range defined by [**new_minA : new_maxA]**. By normalizing the attributes, we speed up the process of learning [8]. In other words, since we measure the distance between each pair of documents for the similarity, normalization prevents the attributes with initially larger range from outweighing attributes with initially smaller range. Different techniques of
normalize the data were applied.

D. Length normalized binary vector with unique words. (Vector-3)

Motivation for length normalized binary vector is length of the document is directly proportional to the probability of words appearance in the document. To nullify this effect the
elements of the vector are divided by the no: of unique words in the document. This vector is termed as *unique words relative binary vector.*

E. Length normalized binary vector with all words. (Vector-4)

In order to have better comprehension of the distribution of words in the documents, normalization of binary vector was done by taking length of documents by counting the words. So it is a variant of the unique words length normalization, where instead of dividing the vector elements by all unique words we divide the elements of the vector by no: of all words in the document thus we get length normalized binary vector, with all words. In addition to the above process of normalization treatment to the binary vector, we adapted few more normalization techniques for frequency vector.

F. Min-Max Normalization . (Vector-5)

Min-max normalization performs a linear transformation on the original data. Suppose that *minA* and *maxA* are the existing minimum and maximum values of an attribute say *A*, Min-max normalization maps a value *v*, to v' in the new range **[new_mainA:new_maxA]** by computing using the equation (1)

$$v' = \frac{v - \min_A}{\max_A - \min_A}(new\_\max_A - new\_\min_A) + new\_\min_A \qquad (1)$$

Min-max normalization preserves the relationships among the original data values. Binary vector is not normalized as the range of the elements in the binary vector is bounded between 0 and 1. So Min_Max normalization is applied only to frequency vector.

G. Min-Max Normalization . (Vector-6)

Similar procedure as that of binary vector normalization, frequency vector normalization is done taking all unique words into consideration. Thus we obtain the length normalized frequency vector with unique words.

H. Length Normalized Frequency vector with all words. (Vector-7)

In the process of obtaining length normalized frequency vector with all words, we divide the frequency vector elements with the number of words in that document. This is so as the size of the document increases the probability of inclusion of (all) the word also increases, so to normalize the increase in the probability, length normalized frequency vector with all words is considered. The vector thus normalized is also called as sum of elements equal to unity
normalization.

I. Root Mean-Square frequency normalizing. (Vector-8)

It is also called as sum of squares normalization, where the numerator represents the frequency of the word in the document while the denominator takes sum of squares of all
the components of the vector, under a square root.

J. Inverse Frequency. (Vector-9)

In case of term frequency feature selection technique, we consider the fact that more a term appears in a document, its weight considered is more than the terms appearing less frequent in the document, in the process of classification, but in case of inverse frequency we go further a step ahead, if a word occurs frequently in many documents its role should be minimized, than the one which appears in certain documents. Inverse frequency feature selection technique takes this fact into consideration. The same can be expressed mathematically

$$IDF(t) = \log \frac{1+|d|}{|dt|} \qquad (2)$$

where |d| is the no: of documents in the corpus, and |dt| is the no: of documents which has the term t.

IV. SIMILARITY MEASURES

Euclidean distance similarity measure is given by the sum of square of the difference of the individual elements [7]. The same can be expressed mathematical as

$$Dist(X,Y) = \sqrt{(x_2-x_1)*(x_2-x_1)+(y_2-y_1)*(y_2-y_1)+....} \qquad (3)$$

Figure-1 shows the results obtained for different k values varying from 10 to 100, and figure-2 gives the results for different values of k varying through 1 to 10.

- Squared Euclidean Distance is similar to the Euclidean distance, but does not have the squareroot over the summation. Figure-3 shows the results obtained for k values varying from 10 to 100, figure-4 gives the results for different values of k varying through 1 to 10. Mathematically square Euclidean distance is expressed as

$$Dist(X,Y) = (x_2-x_1)*(x_2-x_1)+(y_2-y_1)*(y_2-y_1)+... \qquad (4)$$

- Manhattan Distance is a simple similarity measure when compared to the Euclidean and square-Euclidean distance measure, it takes the summation of the absolute difference among the individual elements of the vector. Figure-5 shows the results obtained for k value varying from 10 to 100, and figure-6 gives the results for different values of k varying through 1 to 10. Mathematical expression of Manhattan distance is expressed as

$$Dist(X,Y) = |x_2 - x_1| + |y_2 - y_1| +... \qquad (5)$$

- Chessboard distance is also called as Chebyshev distance, Tchebychev distance), Maximum metric, it is a metric defined on a vector space where the distance between two vectors the greatest of their differences along any coordinate dimension is. It is named after Pafnuty. Figure-7 shows the results obtained for k value varying from 1 to 10, and
  figure -8 gives the results for different values of k varying through 10 to 100. Mathematically the same can be expressed as

$$Dist(X,Y) = Max(|x_1 - y_1|, |x_2 - y_2|, \ldots) \qquad (6)$$

- . Bray Curtis Distance is also called as Sorenson. It is defined as the fraction of absolute difference in the individual elements of the vector to the sum of the individual elements of the two vectors. The same can be expressed mathematically as

$$Dist(X,Y) = \frac{(|x_1 - y_1| + |x_2 - y_2|, \ldots)}{(|x_1 + y_1| + |x_2 + y_2|, \ldots.)} \qquad (7)$$

- Canberra Distance is defined as the ratio of the sum of the absolute difference in the individual elements to the sum of the absolute values of the individual elements in the two vectors. Figure-9 shows the results obtained for k value varying from
  1 to 10, and figure -10 gives the results for different values of k varying through 10 to 100. This is mathematically expressed as

$$Dist(X,Y) = \sum_{k=1}^{n} \frac{|x_{ik} - x_{jk}|}{|x_{ik}| + |x_{kj}|} \qquad (8)$$

IV. K-Nn Implementation

K-Nearest Neighbour algorithm is also called as instance based learning algorithm. Nearest Neighbour classifier are based on learning by analogy, that is by comparing a given
test tuple with training tuples that are similar to it [6]. The training tuples are described by 'n' attributes, in our case attributes are words which are in the lexicon set. Each tuple represents a point in an n-dimension pattern space. When given an unseen tuple, a k-nearest neighbor classifier searches the pattern space for different values of k (which can take any value 1 through some arbitrary number) the training tuples that are closest to the unseen tuple. Depending on the value of k, k training tuples are used which are near to the unseen tuple. Different similarity measures would be applied between two points or tuples say X1 and X2 which have 'n' component elements. It is used to find the similarity (closeness) between the tuples.

For different k tuples, the majority class label is taken, and the unseen tuple class label is declared to be the same as the majority class labels. In case of a tie, arbitrary the tie is resolved. In other words, the test data consists of 2189 documents, the distance between the training and a particular test documents is measured, the class with the nearest training data is taken as the class of the test data, as here K value in K-NN is 1. In case of k value 2 we take 2 smallest distances, and if both belong to same class than the test tuple also belongs to the same class as it is the nearest distance of the training data class, in case of tie an arbitrary consensus is used to resolve the conflict. Based on the similarity between the training and test tuples we obtain confusion matrix which is a good tool for analyzing, how well the classifier can classify the tuples of different classes. A confusion matrix is a plot used to evaluate the performance of a classifier in supervised learning. It is a matrix plot of the predicted versus the actual classes.

For m classes, a confusion matrix is a table of m by m. An entry *CMi,j* in the first m rows and m columns indicates the number of tuples of class *i* that are labeled by the classifier as class *j*. For a classifier to have good accuracy, i,e for a ideal classifier tuples along the diagonal of the confusion matrix would have non-zero values and rest of the elements, very close to zero. Table 1 gives the confusion matrix obtained, in the experiment for different values of k. The table has an additional row and columns to provide totals and recognition rates per class. Finally the accuracy of the classifier can be obtained by taking the sum of truly classified documents divided by the total no of documents which accounts to (566 + 89 + 1069 + 5 + 60 + 55 + 3 + 65) / 2189 = 87.34.

The experiment was repeated for different values of k. Table 2 gives the accuracy of the classifier for varying values of k from 1 through 10 for different vectors which were generated in section 3 of the paper.

We can generate similar tables for other measures also. In the table type-1 vector refers to binary vector, type2 refers to binary vector normalized with number of unique words, type-3 refers to binary vector normalized with number of all words, type 4 refers to frequency vector, type 5 is frequency vector normalized

with number of unique words, type6 vector refers to frequency vector normalized with number of all words, type-7 vector refers to frequency vector normalized with min-max in the range [0:1] and type-8 vector refers to frequency vector normalized with mean square root, vector-9 is the inverse frequency vector respectively. A graph is drawn with K values on X –axis and accuracy of the classifier in percentage on Y axis for all vectors. Figure 3 gives the graph. In order to get a complete insight, the experiment was further carried out for different values of k varying from 1 through 100, we found that as k value increase the accuracy decreases, and for k = 4 with mean square root normalized vector classifier accuracy was found to be maximum, accounting to 94.47% approximately. Figure 4 gives the graph for k values varying from 1 through 100. The accuracy of the classifier was never lower than 55% in all vectors, and for all values of k.

V. Conclusion

The Text classification is very much indispensable in present times as bulk amount of data exist. In order to reuse the generated data classification has to be done. The paper attempts to classify the data by employing different similarity measures, with different vector generation
technique. The experiment conducted depicted as shown in the figures 1 through 10, that the mean square root results are better when compared to all other vectors. More ever we could also find that as the K value increases the classifier accuracy decreases, but for large k values too, we found that the classifier accuracy was never below 55% for different (all) vector types. All the similarity measures gave almost the same results leaving the choice to the users to choose among the different similarity measures.

As a further expansion to the work carried out, we wish to reduce the features and perform the experiments to see how the classifier performs, on feature reduction for
different similarity measures.

References

[1] M.A.Wajeed, T.Adilakshmi "Text Classification Using Machine Learning" Journal of Theoretical and Applied Information Technology Vol. 7 No. 2 Pages 119-123, 2009.

[2] M.A.Wajeed, T.Adilakshmi "Text Classification Using KNN Classifier", International Journal of Computer Science and System Analysis Vol. 3 No. 2, Pages 83-87, July-Dec-2009.

[3] http://www.daviddlewis.com /resources/testcollections/reuters21578/.

[4] http://ai.stanford.edu/~ronnyk/glossary.html.

[5] Padraig Cunnigham and Sarah Jane Delany "K-Nearest Neighbour classifiers" Technical Report UCD-CSI 2007-4.

[6] D. W. Aha, D. Kibler, and M. K. Albert. Instance-based learning algorithms. Machine Learning, 6:37–66, 1991.

[7] Ching-man Au Yeung, Nicholas Gibbins, Nigel Shadbolt, "A k-Nearest-Neighbour Method for Classifying Web Search Results with Data in Folksonomies," wi-iat, vol. 1, pp.70-76, 2008

[8] IEEE/WIC/ACM International Conference on Web Intelligence and Intelligent Agent Technology, 2008.

[9] Data Normalization http://abbottanalytics.blogspot.com/2009/04/why-normalizationmatters-

[10] with-k-means.html

[11] tartarus.org/~martin/PorterStemmer.

[12] Fabrizio Sebastiani. Text classification, automatic. In Keith Brown (ed.), *The Encyclopedia of Language and Linguistics*, 2nd Edition, Vol. 14, Elsevier Science, Amsterdam, NL, 2004.

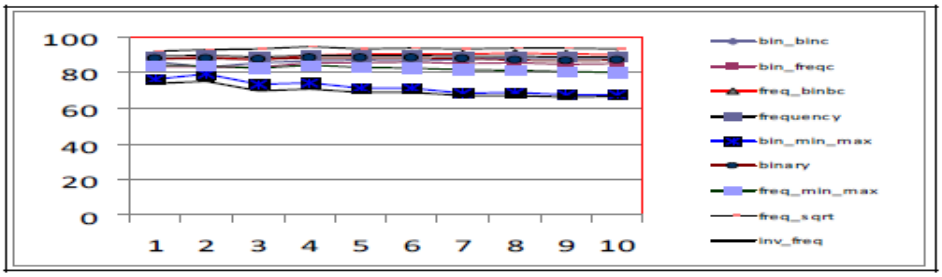

Figure 1 Graph showing accuracy for Euclidean k values 10 to 100

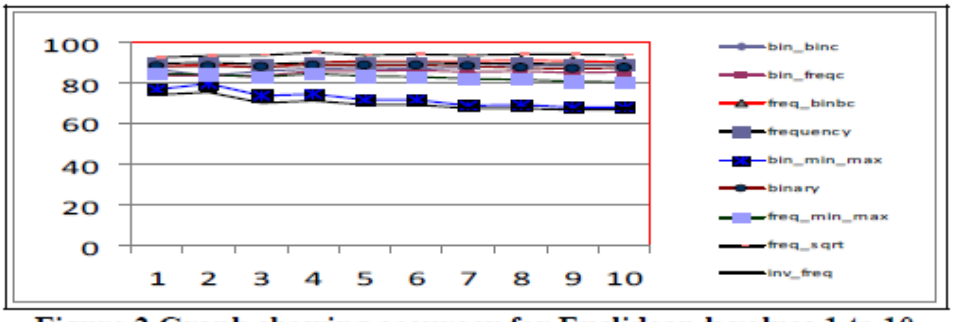

Figure 2 Graph showing accuracy for Euclidean k values 1 to 10

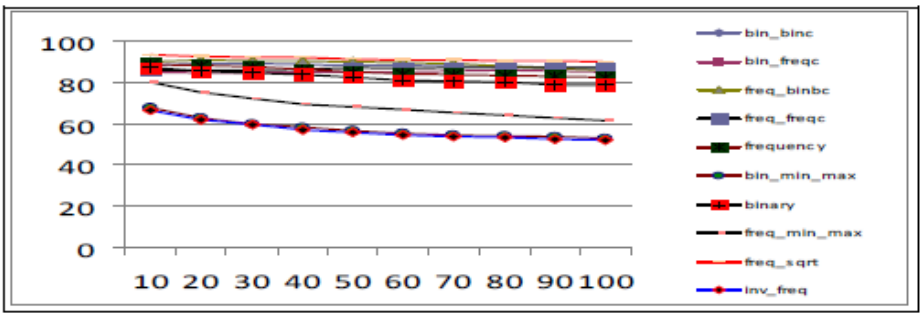

Figure 3 Graph shows accuracy Sqr. Euclidean k value 10 to 100

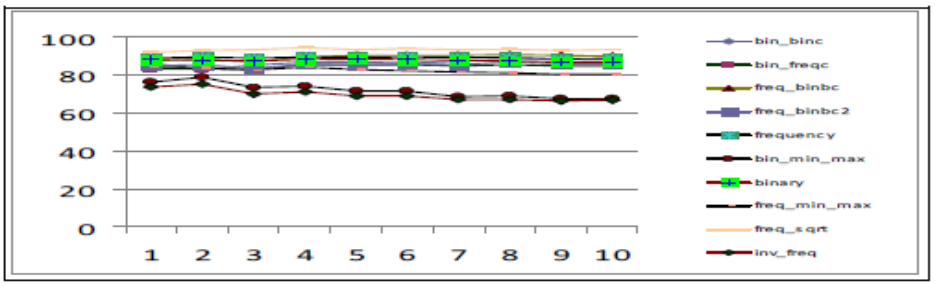

Figure 4 Graph shows accuracy Sqr. Euclidean K values 1 to 10

Table 1. Confusion matrix for k=1 for binary vector

Classes	acq	crude	earn	grain	interest	Money-fx	ship	trade	Total	Recognition
acq	566	5	107	7	3	1	1	7	696	81.32
crude	4	89	15	2	1	0	0	9	121	73.56
earn	11	2	1069	1	0	0	0	0	1083	98.71
grain	0	0	3	5	1	0	0	1	10	50.00
interest	0	0	1	0	60	15	0	5	81	74.07
money-fx	2	2	3	0	17	55	0	8	87	63.21
ship	0	4	4	1	8	0	3	16	36	08.33
trade	0	1	3	2	2	2	0	65	75	86.67
Total	583	103	1205	18	92	73	4	111	2189	87.34

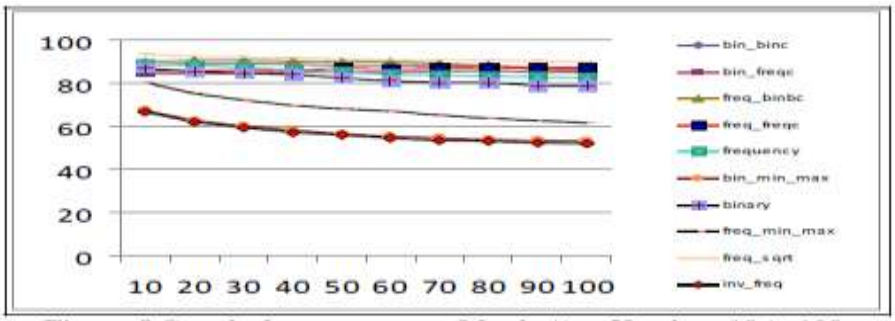

Figure 5 Graph shows accuracy Manhattan K values 10 to 100

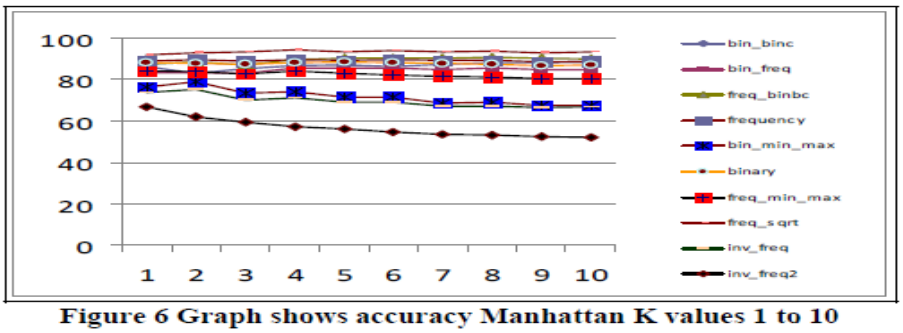

Figure 6 Graph shows accuracy Manhattan K values 1 to 10

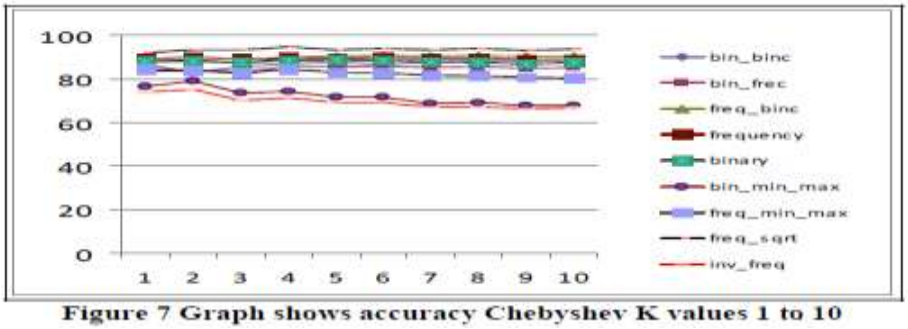

Figure 7 Graph shows accuracy Chebyshev K values 1 to 10

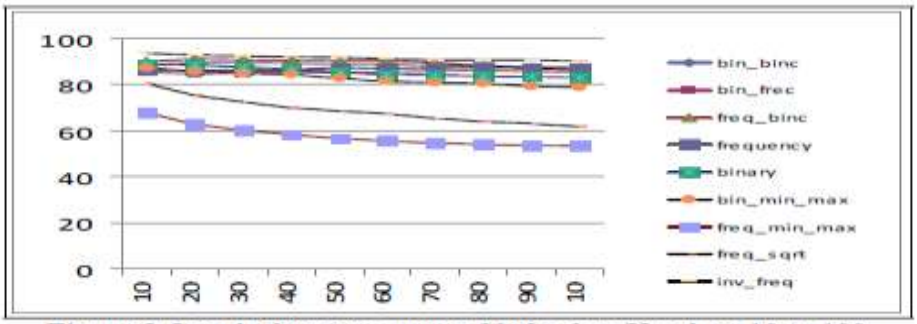

Figure 8 Graph shows accuracy Chebyshev K values 10 to 100

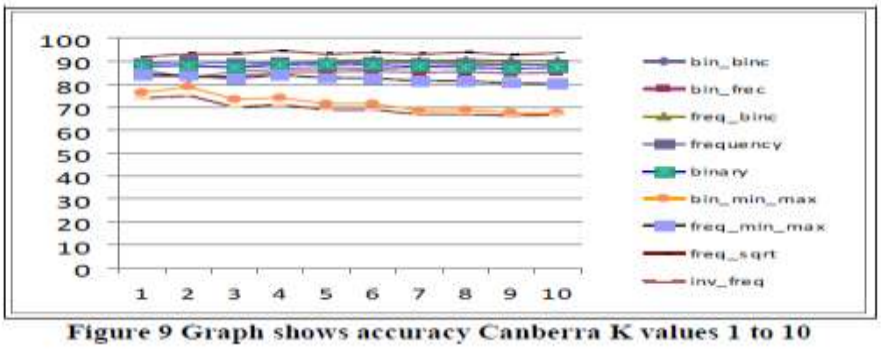

Figure 9 Graph shows accuracy Canberra K values 1 to 10

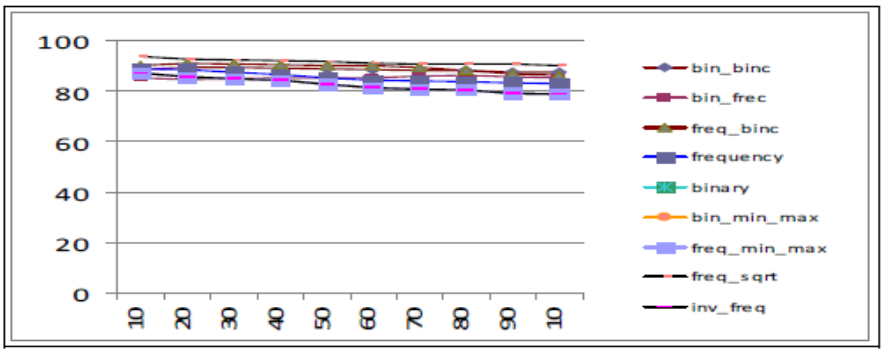

Figure 10 Graph shows accuracy Canberra K values 10 to 100