ISSN:0975-9646

Comparative Analysis of Various Text Classification Algorithms

Varsha C. Pande[1] , Dr. A. S. Khandelwal[2]

[1]Research Scholar,Department of Electronics and Computer Science,RTMNU, Nagpur,Maharashtra, India.
[2]Department of Computer Science, Hislop college, Nagpur,Maharashtra,India.

Abstract-Classification of data has become an important research area. The process of classifying documents into predefined categories based on their content is Text classification. It is the automated assignment of natural language texts to predefined categories. The primary requirement of text retrieval systems is text classification, which retrieve texts in response to a user query, and text understanding systems, which transform text in some way such as answering questions, producing summaries or extracting data. In this paper we are studying the various classification algorithms. Classification is the process of dividing the data to some groups that can act either dependently or independently. Our main aim is to show the comparison of the various classification algorithms like K-nn, Naïve Bayes, Decision Tree, Random Forest and Support Vector Machine (SVM) with rapid miner and find out which algorithm will be most suitable for the users.

Keywords: Text Mining, K-nn, Naïve Bayes, Decision Tree, Random Forest and Support Vector Machine, Rapidminer.

INTRODUCTION

Text mining or knowledge discovery is that sub process of data mining, which is widely being used to discover hidden patterns and significant information from the huge amount of unstructured written material. Text mining is largely growing field of computer science simultaneously to big data and artificial intelligence. Text mining and data mining are similar, except data mining works on structured data while text mining works on semi-structured and unstructured data. Data mining is responsible for extraction of implicit, unknown and potential data and text mining is responsible for explicitly stated data in the given text [1]. Today's world can be described as the digital world as we are being dependent on the digital / electronic form of data. This is environment friendly because we are using very less amount of paper. But again this dependency results in very large amount of data. Even any small activity of human produces electronic data. For example, when any person buys a ticket online, his details are stored in the database. Today approx 80% of electronic data is in the form of text. This huge data is not only unclassified and unstructured (or semi-structured) but also contain useful data, useless data, scientific data and business specific data, etc. According to a survey, 33% of companies are working with very high volume of data i.e. approx. 500TB or more. In this scenario, to extract interesting and previously hidden data pattern process of text mining is used. Commonly, data are stored in the form of text. Broadly there are five steps involved in Text Data Mining. They are:

1. Text Gathering
2. Text Pre-processing
3. Data Analysis (Attribute generation & selection)
4. Visualization (Applying Text Mining algorithms)
5. Evaluation

For this text mining uses techniques of different fields like machine learning, visualization, case-based reasoning, text analysis, database technology statistics, knowledge management, natural language processing and information retrieval [2].

TEXT PRE-PROCESSING

The pre-processing itself is made up of a sequence of steps. The first step in text-pre-processing is the morphological analyses. It is divided into three subcategories: tokenization, filtering and stemming [3].

- TOKENIZATION: Text Mining requires the words and the endings of a document. Finding words and separating them is known as tokenization.
- FILTERING: The next step is filtering of important and relevant words from our list of words which were the output of tokenization. This is also called stop words removal.
- STEMMING: The third step is stemming. Stemming reduces words variants to its root form. Stemming of words increases the recall and precision of the information retrieval in Text Mining. The main idea is to improve recall by automatic handling of word endings by reducing the words to their word roots, at the time of indexing and searching. Stemming is usually done by removing any attached suffixes and prefixes (affixes) from index terms before the actual assignment of the term to the index.

CLASSIFICATION

Classification is a supervised learning technique which places the document according to content. Text classification is largely used in libraries. Text classification or Document categorization has several application such as call center routing, automatic metadata extraction, word sense disambiguation, e-mail forwarding and spam detection, organizing and maintaining large catalogues of Web resources, news articles categorization etc. For text classification many machine learning techniques has been used to evolve rules (which helps to assign particular document to particular category) automatically [1].

Text classification (or text categorization) is the assignment of natural language documents to predefined categories according to their content. Text classification is the act of dividing a set of input documents into two or more classes where each document can be said to belong to one or multiple classes. Huge growth of information flows and especially the explosive growth of Internet promoted growth of automated text classification [4].

CLASSIFICATION METHODS

1. Decision Trees

Decision tree methods rebuild the manual categorization of the training documents by constructing well-defined true/false queries in the form of a tree structure where the nodes represent questions and the leaves represent the corresponding category of documents. After having created the tree, a new document can easily be categorized by putting it in the root node of the tree and let it run through the query structure until it reaches a certain leaf. The main advantage of decision trees is the fact that the output tree is easy to interpret even for persons who are not familiar with the details of the model [5].

2. k-Nearest Neighbor

The categorization itself is usually performed by comparing the category frequencies of the k nearest documents (neighbors). The evaluation of the closeness of documents is done by measuring the angle between the two feature vectors or calculating the Euclidean distance between the vectors. In the latter case the feature vectors have to be normalized to length 1 to take into account that the size of the documents (and, thus, the length of the feature vectors) may differ. A doubtless advantage of the k-nearest neighbor method is its simplicity.

3. Bayesian Approaches

There are two groups of Bayesian approaches in document categorization: Naïve [6] and non-naive Bayesian approaches. The naïve part of the former is the assumption of word independence, meaning that the word order is irrelevant and consequently that the presence of one word does not affect the presence or absence of another one. A disadvantage of Bayesian approaches [7] in general is that they can only process binary feature vectors.

4. Neural Networks

Neural networks consist of many individual processing units called as neurons connected by links which have weights that allow neurons to activate other neurons. Different neural network approaches have been applied to document categorization problems. While some of them use the simplest form of neural networks, known as perceptions, which consist only of an input and an output layer, others build more sophisticated neural networks with a hidden layer between the two others.

The advantage of neural networks is that they can handle noisy or contradictory data very well. The advantage of the high flexibility of neural networks entails the disadvantage of very high computing costs. Another disadvantage is that neural networks are extremely difficult to understand for an average user [4].

5. Vector-based Methods

There are two types of vector-based methods. The centroid algorithm and support vector machines. One of the simplest categorization methods is the centroid algorithm. During the learning stage only the average feature vector for each category is calculated and set as centroid-vector for the category. A new document is easily categorized by finding the centroid-vector closest to its feature vector. The method is also inappropriate if the number of categories is very large. Support vector machines (SVM) need in addition to positive training documents also a certain number of negative training documents which are untypical for the category considered.

An advantage of SVM [8] is its superior runtime-behavior during the categorization of new documents because only one dot product per new document has to be computed. A disadvantage is the fact that a document could be assigned to several categories because the similarity is typically calculated individually for each category.

PERFORMANCE EVALUATION

- **Precision**: exactness – what % of tuples that the classifier labeled as positive are actually positive

  Precision = TP/(TP+FP)
- **Recall:** completeness – what % of positive tuples did the classifier label as positive?

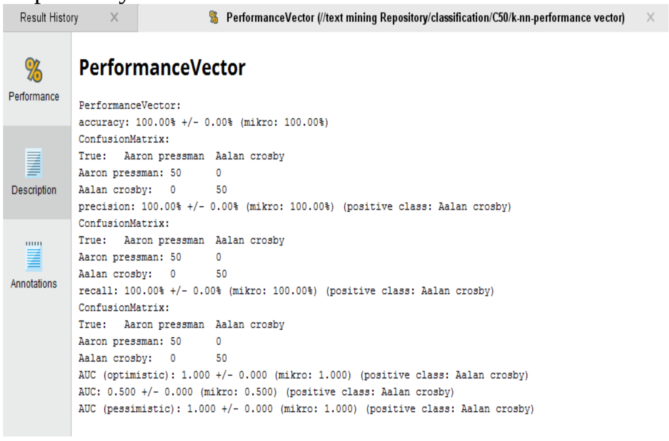
  recall=TP/(TP+FN)
- Perfect score is 1.0.
- Inverse relationship between precision & recall.
- *F* **measure (***F1* or *F***-score)**: harmonic mean of precision and recall,

  F=2×(*precision* × *recall)/ (precision* + *recall)*

IMPLEMENTATION

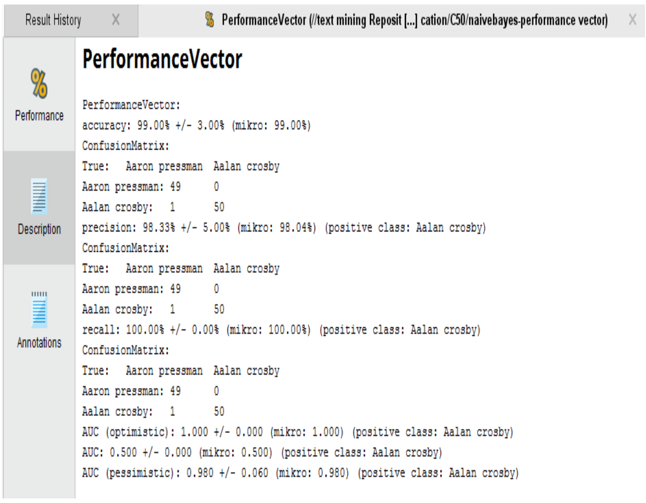
In this study, many classification algorithms have been implemented on two data sets i.e. Tokens dataset and Mini News Group dataset both are publically available datasets, And the performance of this algorithm has been analyzed by the Text Mining tool RAPIDMINER.

We have applied five algorithms i.e. **K-NN, Naïve Bayes, Decision Tree, Random Forest and Support Vector Machine (SVM)** in **C-50 dataset** and the results are shows in figure1, figure2, figure3, figure4 and figure5 respectively.


Result History
PerformanceVector (//text mining Repository/classification/C50/k-nn-performance vector)
Performance
Description
Annotations
PerformanceVector

```
PerformanceVector:
accuracy: 100.00% +/- 0.00% (mikro: 100.00%)
ConfusionMatrix:
True:   Aaron pressman  Aalan crosby
Aaron pressman: 50      0
Aalan crosby:   0       50
precision: 100.00% +/- 0.00% (mikro: 100.00%) (positive class: Aalan crosby)
ConfusionMatrix:
True:   Aaron pressman  Aalan crosby
Aaron pressman: 50      0
Aalan crosby:   0       50
recall: 100.00% +/- 0.00% (mikro: 100.00%) (positive class: Aalan crosby)
ConfusionMatrix:
True:   Aaron pressman  Aalan crosby
Aaron pressman: 50      0
Aalan crosby:   0       50
AUC (optimistic): 1.000 +/- 0.000 (mikro: 1.000) (positive class: Aalan crosby)
AUC: 0.500 +/- 0.000 (mikro: 0.500) (positive class: Aalan crosby)
AUC (pessimistic): 1.000 +/- 0.000 (mikro: 1.000) (positive class: Aalan crosby)
```

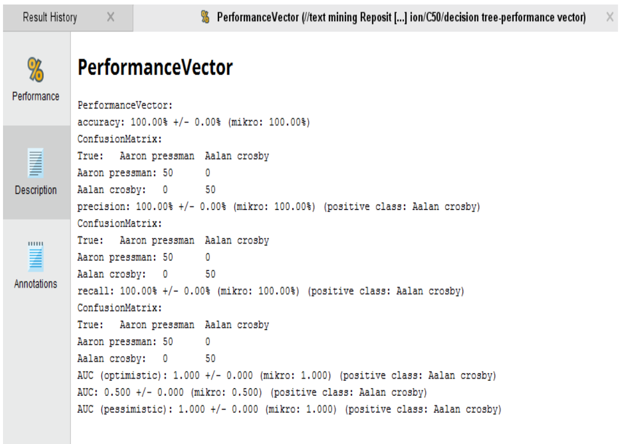
Fig1: K-NN algorithm on C-50 dataset.


Result History
X
PerformanceVector (//text mining Reposit [...] cation/C50/naivebayes-performance vector)
X
Performance
Description
Annotations

```
PerformanceVector

PerformanceVector:
accuracy: 99.00% +/- 3.00% (mikro: 99.00%)
ConfusionMatrix:
True:   Aaron pressman  Aalan crosby
Aaron pressman: 49      0
Aalan crosby:   1       50
precision: 98.33% +/- 5.00% (mikro: 98.04%) (positive class: Aalan crosby)
ConfusionMatrix:
True:   Aaron pressman  Aalan crosby
Aaron pressman: 49      0
Aalan crosby:   1       50
recall: 100.00% +/- 0.00% (mikro: 100.00%) (positive class: Aalan crosby)
ConfusionMatrix:
True:   Aaron pressman  Aalan crosby
Aaron pressman: 49      0
Aalan crosby:   1       50
AUC (optimistic): 1.000 +/- 0.000 (mikro: 1.000) (positive class: Aalan crosby)
AUC: 0.500 +/- 0.000 (mikro: 0.500) (positive class: Aalan crosby)
AUC (pessimistic): 0.980 +/- 0.060 (mikro: 0.980) (positive class: Aalan crosby)
```

Fig2: Naïve Bayes algorithm on C-50 dataset

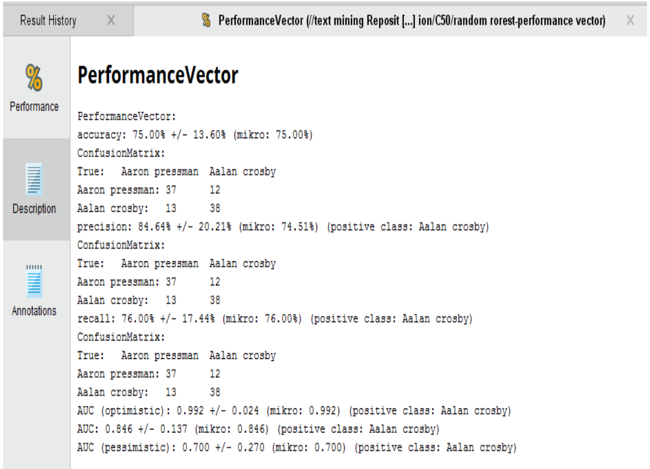

Result History
X
PerformanceVector (//text mining Reposit [...] ion/C50/decision tree-performance vector)
X
Performance
Description
Annotations

```
PerformanceVector

PerformanceVector:
accuracy: 100.00% +/- 0.00% (mikro: 100.00%)
ConfusionMatrix:
True:   Aaron pressman  Aalan crosby
Aaron pressman: 50      0
Aalan crosby:   0       50
precision: 100.00% +/- 0.00% (mikro: 100.00%) (positive class: Aalan crosby)
ConfusionMatrix:
True:   Aaron pressman  Aalan crosby
Aaron pressman: 50      0
Aalan crosby:   0       50
recall: 100.00% +/- 0.00% (mikro: 100.00%) (positive class: Aalan crosby)
ConfusionMatrix:
True:   Aaron pressman  Aalan crosby
Aaron pressman: 50      0
Aalan crosby:   0       50
AUC (optimistic): 1.000 +/- 0.000 (mikro: 1.000) (positive class: Aalan crosby)
AUC: 0.500 +/- 0.000 (mikro: 0.500) (positive class: Aalan crosby)
AUC (pessimistic): 1.000 +/- 0.000 (mikro: 1.000) (positive class: Aalan crosby)
```

Fig3: Decision tree algorithm on C-50 dataset.

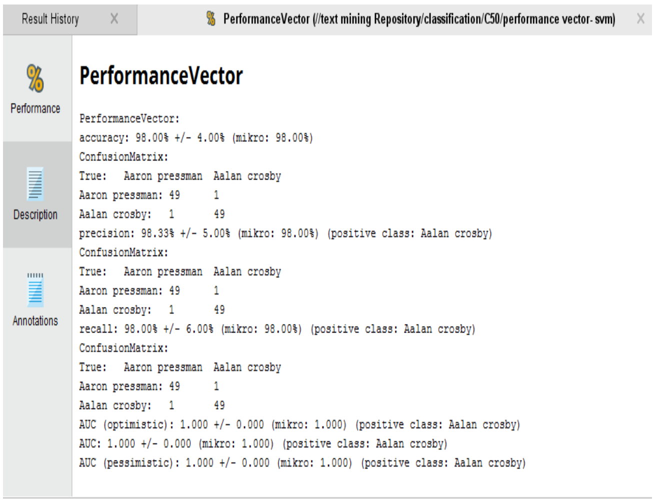
Similarly, we have applied five algorithms i.e. **K-NN, Naïve Bayes, Decision Tree, Random Forest and Support Vector Machine (SVM)** on **gutenberg dataset** and the results are shown in figure6(K-NN), figure7(**Naïve Bayes)**, figure8(**Decision Tree)**, and figure9(**SVM)**.


Result History
X
PerformanceVector (//text mining Reposit [...] ion/C50/random rorest-performance vector)
X
Performance
Description
Annotations

```
PerformanceVector

PerformanceVector:
accuracy: 75.00% +/- 13.60% (mikro: 75.00%)
ConfusionMatrix:
True:   Aaron pressman  Aalan crosby
Aaron pressman: 37      12
Aalan crosby:   13      38
precision: 84.64% +/- 20.21% (mikro: 74.51%) (positive class: Aalan crosby)
ConfusionMatrix:
True:   Aaron pressman  Aalan crosby
Aaron pressman: 37      12
Aalan crosby:   13      38
recall: 76.00% +/- 17.44% (mikro: 76.00%) (positive class: Aalan crosby)
ConfusionMatrix:
True:   Aaron pressman  Aalan crosby
Aaron pressman: 37      12
Aalan crosby:   13      38
AUC (optimistic): 0.992 +/- 0.024 (mikro: 0.992) (positive class: Aalan crosby)
AUC: 0.846 +/- 0.137 (mikro: 0.846) (positive class: Aalan crosby)
AUC (pessimistic): 0.700 +/- 0.270 (mikro: 0.700) (positive class: Aalan crosby)
```

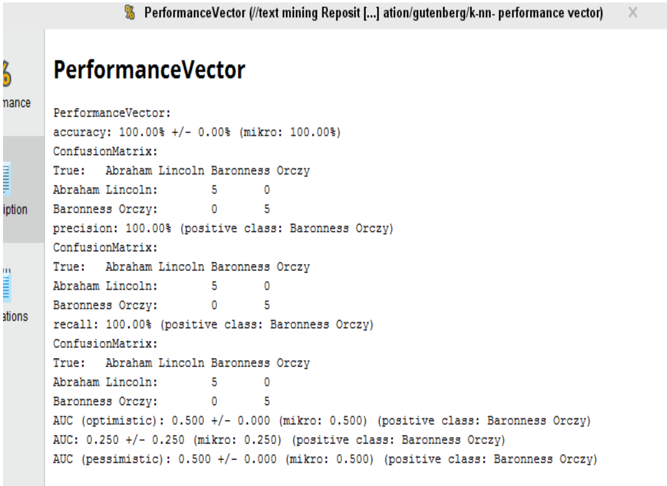
Fig4: Random Forest algorithm on C-50 dataset


Result History
X
PerformanceVector (//text mining Repository/classification/C50/performance vector- svm)
X
Performance
Description
Annotations

```
PerformanceVector

PerformanceVector:
accuracy: 98.00% +/- 4.00% (mikro: 98.00%)
ConfusionMatrix:
True:   Aaron pressman  Aalan crosby
Aaron pressman: 49      1
Aalan crosby:   1       49
precision: 98.33% +/- 5.00% (mikro: 98.00%) (positive class: Aalan crosby)
ConfusionMatrix:
True:   Aaron pressman  Aalan crosby
Aaron pressman: 49      1
Aalan crosby:   1       49
recall: 98.00% +/- 6.00% (mikro: 98.00%) (positive class: Aalan crosby)
ConfusionMatrix:
True:   Aaron pressman  Aalan crosby
Aaron pressman: 49      1
Aalan crosby:   1       49
AUC (optimistic): 1.000 +/- 0.000 (mikro: 1.000) (positive class: Aalan crosby)
AUC: 1.000 +/- 0.000 (mikro: 1.000) (positive class: Aalan crosby)
AUC (pessimistic): 1.000 +/- 0.000 (mikro: 1.000) (positive class: Aalan crosby)
```

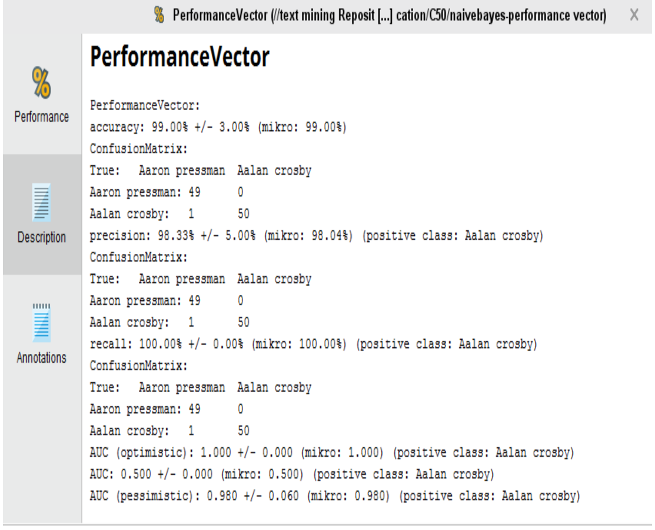
Fig5: SVM algorithm on C-50 dataset


PerformanceVector (//text mining Reposit [...] ation/gutenberg/k-nn- performance vector)
X
nance
iption
ations

```
PerformanceVector

PerformanceVector:
accuracy: 100.00% +/- 0.00% (mikro: 100.00%)
ConfusionMatrix:
True:   Abraham Lincoln Baronness Orczy
Abraham Lincoln:        5       0
Baronness Orczy:        0       5
precision: 100.00% (positive class: Baronness Orczy)
ConfusionMatrix:
True:   Abraham Lincoln Baronness Orczy
Abraham Lincoln:        5       0
Baronness Orczy:        0       5
recall: 100.00% (positive class: Baronness Orczy)
ConfusionMatrix:
True:   Abraham Lincoln Baronness Orczy
Abraham Lincoln:        5       0
Baronness Orczy:        0       5
AUC (optimistic): 0.500 +/- 0.000 (mikro: 0.500) (positive class: Baronness Orczy)
AUC: 0.250 +/- 0.250 (mikro: 0.250) (positive class: Baronness Orczy)
AUC (pessimistic): 0.500 +/- 0.000 (mikro: 0.500) (positive class: Baronness Orczy)
```

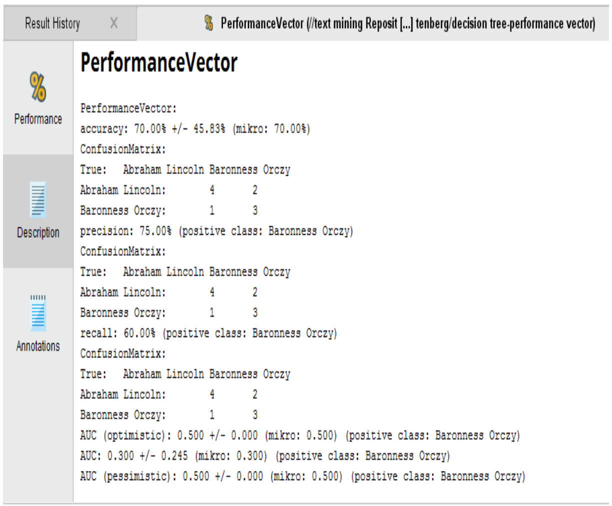
Fig6: K-NN algorithm on Gutenberg dataset


PerformanceVector (//text mining Reposit [...] cation/C50/naivebayes-performance vector)
X
Performance
Description
Annotations

```
PerformanceVector

PerformanceVector:
accuracy: 99.00% +/- 3.00% (mikro: 99.00%)
ConfusionMatrix:
True:   Aaron pressman  Aalan crosby
Aaron pressman: 49      0
Aalan crosby:   1       50
precision: 98.33% +/- 5.00% (mikro: 98.04%) (positive class: Aalan crosby)
ConfusionMatrix:
True:   Aaron pressman  Aalan crosby
Aaron pressman: 49      0
Aalan crosby:   1       50
recall: 100.00% +/- 0.00% (mikro: 100.00%) (positive class: Aalan crosby)
ConfusionMatrix:
True:   Aaron pressman  Aalan crosby
Aaron pressman: 49      0
Aalan crosby:   1       50
AUC (optimistic): 1.000 +/- 0.000 (mikro: 1.000) (positive class: Aalan crosby)
AUC: 0.500 +/- 0.000 (mikro: 0.500) (positive class: Aalan crosby)
AUC (pessimistic): 0.980 +/- 0.060 (mikro: 0.980) (positive class: Aalan crosby)
```

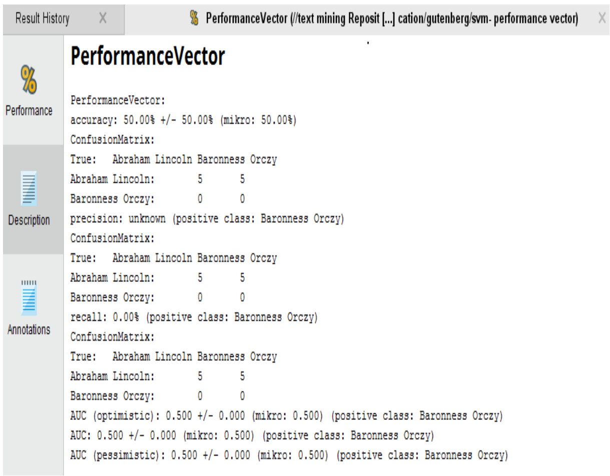
Fig7: Naïve Bayes algorithm on Gutenberg dataset


Result History X
PerformanceVector (//text mining Reposit [...] tenberg/decision tree-performance vector)
Performance
Description
Annotations

PerformanceVector

```
PerformanceVector:
accuracy: 70.00% +/- 45.83% (mikro: 70.00%)
ConfusionMatrix:
True:   Abraham Lincoln Baronness Orczy
Abraham Lincoln:        4       2
Baronness Orczy:        1       3
precision: 75.00% (positive class: Baronness Orczy)
ConfusionMatrix:
True:   Abraham Lincoln Baronness Orczy
Abraham Lincoln:        4       2
Baronness Orczy:        1       3
recall: 60.00% (positive class: Baronness Orczy)
ConfusionMatrix:
True:   Abraham Lincoln Baronness Orczy
Abraham Lincoln:        4       2
Baronness Orczy:        1       3
AUC (optimistic): 0.500 +/- 0.000 (mikro: 0.500) (positive class: Baronness Orczy)
AUC: 0.300 +/- 0.245 (mikro: 0.300) (positive class: Baronness Orczy)
AUC (pessimistic): 0.500 +/- 0.000 (mikro: 0.500) (positive class: Baronness Orczy)
```

Fig8: Decision tree algorithm on Gutenberg dataset


Result History X
PerformanceVector (//text mining Reposit [...] cation/gutenberg/svm- performance vector) X
Performance
Description
Annotations

PerformanceVector

```
PerformanceVector:
accuracy: 50.00% +/- 50.00% (mikro: 50.00%)
ConfusionMatrix:
True:   Abraham Lincoln Baronness Orczy
Abraham Lincoln:        5       5
Baronness Orczy:        0       0
precision: unknown (positive class: Baronness Orczy)
ConfusionMatrix:
True:   Abraham Lincoln Baronness Orczy
Abraham Lincoln:        5       5
Baronness Orczy:        0       0
recall: 0.00% (positive class: Baronness Orczy)
ConfusionMatrix:
True:   Abraham Lincoln Baronness Orczy
Abraham Lincoln:        5       5
Baronness Orczy:        0       0
AUC (optimistic): 0.500 +/- 0.000 (mikro: 0.500) (positive class: Baronness Orczy)
AUC: 0.500 +/- 0.000 (mikro: 0.500) (positive class: Baronness Orczy)
AUC (pessimistic): 0.500 +/- 0.000 (mikro: 0.500) (positive class: Baronness Orczy)
```

Fig9: SVM algorithm on Gutenberg dataset

The following table shows the results for C-50 Dataset and Gutenberg Dataset.

Table 1: Results For C-50 Dataset

Algorithm	Accuracy	Precision	Recall	Execution Time
K-nn	100	100	100	15
Naïve bayes	99	98	100	14
Decision Tree	100	100	100	20
Random Forest	75	84.64	76	141.6
SVM	98	98.33	98	91.8

Note: The values of Accuracy, Precision, and Recall are in percentage and the execution time is in **seconds**.

Table 2: Results For Gutenberg Dataset

Algorithm	Accuracy	Precision	Recall	Execution Time
K-nn	100	100	100	395
Naïve bayes	100	100	100	480
Decision Tree	70	75	60	471
Random Forest	-	-	-	-
SVM	50	unknown	0	276

The Random Forest algorithm on Gutenberg dataset does not show the results for Accuracy, Precision, Recall and execution time it shows "memory exceeds" error.

CONCLUSION

Text mining techniques are mainly used in medicals, banking, insurances, education etc. The classification algorithms K-NN, Naïve Bayes, Decision Tree, Random Forest and SVM have their own importance and we use them on the behavior of the two datasets that are C-50 and Gutenberg, but on the basis of this research we found that K-NN classification algorithm is simplest algorithm as compared to other algorithms.

The different classification algorithms are studied and implemented using **RAPIDMINER Studio 7.5.003**. The implementation results show the values for Accuracy, Precision, Recall and execution time. The overall results for the all the algorithms are shown in Table1 and Table 2. From the results it is clear that, the K-NN algorithm is better than other algorithms on both the datasets.

The Overall Performance of all the algorithms is better for **C-50** rather than **Gutenberg dataset.**

REFERENCES

1) Abhishek Kaushik and Sudanshu Naithani "A Comprehensive study of Text Mining Approach".

2) Vishal Gupta And Gurpreet S. Lehal "A Survey of Text Mining Techniques and Application".

3) Ms. Anjali Ganesh Jivani, Prof. B. S. Parekh "A Comparative Study of Text Data Mining Algorithms and its Applications".

4) S.Niharika,V.SnehaLatha and D.R.Lavanya"A Survey On Text Categorization".

5) D. E. Johnson, F. J. Oles, T. Zhang, T. Goetz,"A decision-tree-based symbolic rule induction system for text categorization", IBM Systems Journal, September 2002.

6) Kim S. B., Rim H. C., Yook D. S. and Lim H. S., "Effective Methods for Improving Naïve Bayes Text Classifiers", LNAI 2417, 2002, pp.414-423.

7) Klopotek M. and Woch M., "Very Large Bayesian Networks in Text Classification", ICCS 2003, LNCS 2657, 2003, pp. 397-406.

8) Joachims, T., Transductive inference for text classification using support vector machines. Proceedings of ICML-99, 16th International Conference on Machine Learning, eds. I. Bratko & S. Dzeroski, Morgan Kaufmann Publishers, San Francisco, US: Bled, SL, ,1999 ,pp. 200–209.