**DATA MINING ASSIGNMENT 2**

**TASK 1:**

**Describe which question you will be investigating and why you think your choice is an interesting question to investigate.**

Two important applications of text mining are informational retrieval and document search. Finding out the relevant words out of many irrelevant words in the dataset and performing mining on those relevant words will definitely result in good outcomes in text mining. TF-IDF is invented to perform the information retrieval and document search. It increases proportionally to the frequency of a word in a document, but is offset by the frequency of documents that contain the particular word. So the common words in every document such as the, is, as, etc. are ranked low since they don't mean much to that particular document.

I would like to investigate the impact of TF-IDF score on the performance of classifiers. So the question is “Is TF-IDF score beneficial to improve the performance of classifier?

**TASK 2:**

**Convert the text dataset into TWO different databases in ARFF format, based on your chosen question. Explain the conversion techniques and parameters that you have used, along with any other pre-processing you wish to do.**

The text dataset is converted into two different databases in ARFF format. The Weka needs the data to be converted in ARFF format to perform different classification tasks. The dataset is loaded in ARFF format using text directory loader where each text files in the directory are loaded as a string attribute and are labelled using the subdirectory names.

The data pre-processing is performed by setting some parameters in the filter:

* The attribute selection mode ‘invertSelection’ is false so it classification is performed only on selected attributes.
* LowercaseTokens is set True, sets all text to lower case characters.
* The minTermFreq parameter is set to 1 so that the word that appears atleast one time in the text string will be added in the attributes list.
* The normalizeDocLength is set as ‘No normalization’ as our investigation is based word frequency in a document and do not want to normalize the word frequencies for a document.
* The periodicPruning is set as -1.0 rate to periodically prune the dictionary.
* The stopwords is set as weka. So the list of stopwords are defined by Weka.
* The UseStopList option is set to true so that the stopwords in the file defined by Weka are removed from the database.
* The doNotOperateOnPerClassBasis parameter is set to False. If it is set to True, the maximum number of words and the minimum term frequency is not enforced on a per-class basis but based on the documents in all the classes. The analysis is based on the classification algorithms’ performance, the focus is on per-class basis.
* The LovinsStemmer algorithm is selected as stemmer so that the words are reduced to their roots.
* The OutputWordCount is set true which indicates the word frequency as real numbers.
* WordTokenizer is selected as tokenizer to split the string into tokens.
* WordtoKeep is set to value 1000. This would improve the complexity of the model.
* The characters that are not considered as words are removed from the attribute list by using remove option.

The dataset is saved as two different databases for performing the analysis. Database 1: Perform the Classification based on TF-IDF score.

The *Term Frequency* and *Inverse document frequency* options in StringToWordVector filter are set to True. The Term frequency is the count of that term in a document and the inverse document frequency is the measure of information the word provides across all documents in the directory.TF-IDF score is a numerical value that indicates the importance of word in a document.

Database 2: Perform the Classification based on word frequency.

The *Term Frequency* and *Inverse document frequency* options in StringToWordVector filter is set to False.

All other preprocessing parameters are set same as described above in both databases. There are 911 instances and 1569 attributes in Database 1 and in Database 2.

**TASK 3:**

**For each database, produce a table and a graph of classification performance against training set size for the following three classifiers: decision-tree (J48), Naïve Bayes, Support Vector Machine. For the Support-Vector Machine you must determine the kernel and its parameters.**

The performance is evaluated based on the ***accuracy*** score of three classifiers*: Decision-tree (J48), Naïve Bayes, Support Vector Machine (SVM).* The data is split and fed to the classifier to evaluate their performance.

The databases are split at 10%, 25%, 50%, 75% and 90%. The performance matrix along with the total number of instances, correctly and incorrectly classified instances for each split of each database is shown in separate tables. The accuracy values of database 1 and database 2 are added in a table and the performance is plotted in a graph for each classifier.

1. ***Decision-tree (J48)***

The J48 classifier is used for classification with its default parameters.

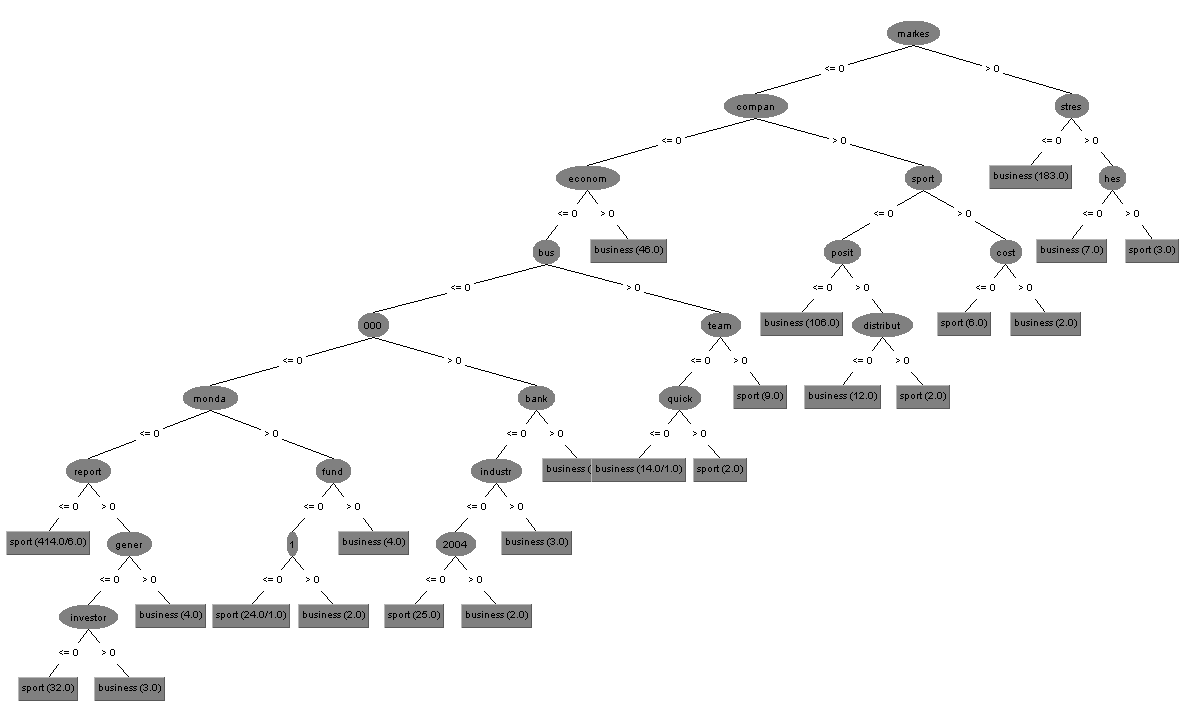


Figure 3: Decision Tree

1. Database 1:

|  |  |  |  |
| --- | --- | --- | --- |
| Percentage split | Total Number of Instances | Number of Correctly classified instances | Number of Incorrectly classified instances |
| 10% | 820 | 757 | 63 |
| 25% | 683 | 638 | 45 |
| 50% | 455 | 426 | 29 |
| 75% | 228 | 220 | 8 |
| 90% | 91 | 87 | 4 |

Table 3.1.1: Performance of J48 classifier on Database 1

1. Database 2:

|  |  |  |  |
| --- | --- | --- | --- |
| Percentage split | Total Number of Instances | Number of Correctly classified instances | Number of Incorrectly classified instances |
| 10% | 820 | 757 | 63 |
| 25% | 683 | 638 | 45 |
| 50% | 455 | 426 | 29 |
| 75% | 228 | 220 | 8 |
| 90% | 91 | 87 | 4 |

Table 3.1.2: Performance of J48 classifier on Database 2

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| J48 | 10 % | 25 % | 50 % | 75 % | 90 % |
| Database 1 | 92.68% | 94.43 % | 93.62 % | 96.49% | 95.60% |
| Database 2 | 92.31 % | 93.41 % | 93.62 % | 96.49% | 95.60% |

Table 3.1.3: Accuracy scores of Decision-tree(J48) classifier

The Accuracy scores of decision tree is plotted using line graph.

Figure 3.1: Decision-tree (J48) classifier

1. ***Naïve Bayes***

Naïve Bayes classifier perform classification based on Bayes Theorem. The default parameters are used to run the classifier and display the performance values.

1. Database 1:

|  |  |  |  |
| --- | --- | --- | --- |
| Percentage split | Total Number of Instances | Number of Correctly classified instances | Number of Incorrectly classified instances |
| 10% | 820 | 811 | 9 |
| 25% | 683 | 674 | 9 |
| 50% | 455 | 452 | 3 |
| 75% | 228 | 226 | 2 |
| 90% | 91 | 91 | 0 |

Table 3.1.1: Performance of Naïve Bayes classifier on Database 1

1. Database 2:

|  |  |  |  |
| --- | --- | --- | --- |
| Percentage split | Total Number of Instances | Number of Correctly classified instances | Number of Incorrectly classified instances |
| 10% | 820 | 808 | 12 |
| 25% | 683 | 671 | 12 |
| 50% | 455 | 450 | 5 |
| 75% | 228 | 226 | 2 |
| 90% | 91 | 91 | 0 |

Table 3.1.2: Performance of Naïve Bayes classifier on Database 2

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Naïve Bayes | 10 % | 25 % | 50 % | 75 % | 90 % |
| Database 1 | 98.90% | 98.68% | 99.54% | 99.12% | 100% |
| Database 2 | 98.56% | 98.24% | 98.90% | 99.12% | 100 % |

Table 3.2: Accuracy scores of Naïve Bayes classifier

The Performance of Naïve Bayes classifier based on accuracy for database 1 and database 2 is plotted using a line graph as shown below.

Figure 3.2: Naïve Bayes classifier

1. ***Support Vector Machine***

Support Vector Machine (SVM) classification is performed using LibSVM classifier. It has different parameters.

* The type of SVM chose is C-SVC (classification).
* The C parameter which denotes the error cost for C-SVC is set to 10.0.
* The degree of kernel is set to 2.
* The gamma parameter which sets the complexity of decision surface is set to 0.03.
* The normalize parameter is set True. It converts all numerical values to same range of values.
* The kernelType parameter is set as Radial basis function (RBF).
* All other parameters are set as default values.

1. Database 1:

|  |  |  |  |
| --- | --- | --- | --- |
| Percentage split | Total Number of Instances | Number of Correctly classified instances | Number of Incorrectly classified instances |
| 10% | 820 | 786 | 34 |
| 25% | 683 | 674 | 9 |
| 50% | 455 | 451 | 4 |
| 75% | 228 | 227 | 1 |
| 90% | 91 | 91 | 0 |

Table 3.1.1: Performance of SVM classifier on Database 1

1. Database 2:

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Percentage split | Total Number of Instances | Number of Correctly classified instances | Number of Incorrectly classified instances | Confusion matrix  a = business  b = sport |
| 10% | 820 | 707 | 113 | |  |  |  | | --- | --- | --- | | **a** | **b** |  | | **353** | **6** | **a** | | **107** | **354** | **b** | |
| 25% | 683 | 654 | 29 | |  |  |  | | --- | --- | --- | | **a** | **b** |  | | **282** | **6** | **a** | | **23** | **372** | **b** | |
| 50% | 455 | 443 | 12 | |  |  |  | | --- | --- | --- | | **a** | **b** |  | | **184** | **2** | **a** | | **10** | **259** | **b** | |
| 75% | 228 | 220 | 8 | |  |  |  | | --- | --- | --- | | **a** | **b** |  | | **90** | **0** | **a** | | **8** | **130** | **b** | |
| 90% | 91 | 90 | 1 | |  |  |  | | --- | --- | --- | | **a** | **b** |  | | **35** | **0** | **a** | | **1** | **55** | **b** | |

Table 3.1.1: Performance of SVM classifier on Database 1

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| SVM | 10 % | 25 % | 50 % | 75 % | 90 % |
| Database 1 | 97.43% | 98.97% | 98.90% | 99.56% | 100% |
| Database 2 | 86.21% | 95.75% | 97.17 % | 96.49% | 98.90% |

Table 3.3: Performance of Support Vector Machine classifier

The Performance of SVM classifier on database 1 and database 2 is plotted using a line graph with x-axis having training data size and y-axis having accuracy scores is shown below.

Figure 3.3: SVM classifier

**TASK 4:**

**Write a conclusion. You should at least compare the performance of the different learning algorithms on your databases and answer the question you posed in part (1).**

The Decision Tree classifier classifies both the databases and the performance matrix have similar values for both. The Naïve Bayes classifier performs well on both, with an accuracy above 98% increasing gradually to 100% as the training size increased. The Support Vector Machine (SVM) classifier performs very well on both databases. The accuracy of database 1 is higher that database 2. The accuracy of classifier reach 100% at 90% split for database 1

The accuracy scores of all classifiers in both databases are compared to reach the conclusion. The database using TF-IDF performs very well on low training data sets compared to other database. Naive Bayes classifier performs well than other. Thus the database transformed using TF-IDF Transform produce better results than database without TF-IDF Transform. Thus, concludes that the TF-IDF score improves the performance of classifier.