Report : IR_A2_Q2

Objective - To calculate the tf-icf score for each class in the dataset and then implement the naive bayes model to check the accuracy of the classifier.

Importing the dataset - Used unzip method to unzip the csv file and then read the file using pandas dataframe.

Preprocessing:
- Each text in the text column in lower cased.
- Then we remove punctuations using regex.
- Then we tokenize and remove stopwords using nltk stopwords library.
- Lemmatize the tokenized arrays of entire column .
- Finally store strings instead of tokenized array in the dataset.
- Dropped article_id column as it was of no use for our classifier.
- We find out the tf-icf weights of all the entries in the dataset using tf-icf function which returns Matrix of tf-icf weights . tf-ic arrayf= [no of rows][no of features].

Methodology:
- We split the dataset into train and test using X=tf-icf matrix and Y=df['category'] in ratio 70:30.
- Using Multinomial naive bayes classifier of sklearn , trained the model using X_train and y_train.
- Calculated the performance of training and testing set using acc, prec, recall and f1-score.
- We also calculated the probability of each class using freq of docs in each class .
- We find out prob of each feature in the tf-idf matrix using feature_probabilities[category][j]=category_feature_count[j] / category_feature_count.sum() for jth feature.

Improving the classifier:
- Tested with different parameters :
  - Different train-test split ratio,
  - Gaussian-Naive_Bayes,
  - Diff alpha values.
  - Using tf-icf-idf weights
  - Using tf-idf weights.

Results And Conclusion :
- Train-Test Split = 70:30
  - For tf-icf
  - Training Accuracy : 0.9961649089165868
  - Testing Accuracy: 0.9686800894854586
  - Precision: 0.9691648861472719
  - Recall: 0.9686800894854586
  - F1-score: 0.9687540264721474

For tf-icf-idf
Testing Accuracy: 0.9798657718120806
Precision: 0.9800668875497954
Recall: 0.9798657718120806
F1-score: 0.9799166342168256

For tf-idf
Testing Accuracy: 0.970917225950783
Precision: 0.9712988371053634
Recall: 0.970917225950783
F1-score: 0.9708945788234244

- Train-Test Split = 80:20
  For tf-icf
  Training Accuracy : 0.9949664429530202
  Testing Accuracy: 0.9865771812080537
  Precision: 0.9867902418237988
  Recall: 0.9865771812080537
  F1-score: 0.9865691627977836

- Train-Test Split = 60:40
  For tf-icf
  Training Accuracy : 0.9955257270693513
  Testing Accuracy: 0.9781879194630873
  Precision: 0.9785968828485272
  Recall: 0.9781879194630873
  F1-score: 0.9782360124933803

- Train-Test Split = 50:50
  For tf-icf
  Training Accuracy : 0.9959731543624161
  Testing Accuracy: 0.9771812080536912
  Precision: 0.9779968246356023
  Recall: 0.9771812080536912
  F1-score: 0.9772372749337948

With diff parameters :
  With Gaussian Naive Bayes
  Testing Accuracy: 0.901565995525727
  Precision: 0.9035883148610421
  Recall: 0.901565995525727
  F1-score: 0.9017096526219379

With alpha = 0.6

```
Testing Accuracy: 0.9686800894854586
Precision: 0.9687905519919578
Recall: 0.9686800894854586
F1-score: 0.9686742809864403
```

With Tf-idf-icf

```
Testing Accuracy: 0.9798657718120806
Precision: 0.9800668875497954
Recall: 0.9798657718120806
F1-score: 0.9799166342168256
```

From above results, we can see that for train-test ratio = 0.2 , accuracy is maximum . i.e. 98.65 %.
With Gaussian naive bayes, acc is decreasing drastically to 90% , hence multinomial naive bayes is better our our dataset. It is because multinomial NB considers no of occurrences of a word in the data while gaussian NB is used for continuous and normally distributed dataset.

Decreasing alpha parameter to 0.6 slightly increases the accuracy to 96.96 % from 96.6% .
With Tf-idf-icf values performance of the classifier increases slightly.

Prob of each category

```
{'sport': 0.22722914669223393,
 'entertainment': 0.18120805369127516,
 'politics': 0.19367209971236818,
 'business': 0.22722914669223393,
 'tech': 0.17066155321188878}
```

Probability of each feature in tf-icf matrix


```
{'business': {0: 0.0,
  1: 0.0,
  2: 0.0,
  3: 6.0232497440118856e-05,
  4: 0.0,
  5: 0.0,
  6: 0.0,
  7: 0.0,
  8: 2.0077499146706285e-05,
  9: 4.015499829341257e-05,
  10: 4.015499829341257e-05,
  11: 0.0,
  12: 0.00038147248378741946,
  13: 0.0,
  14: 0.0,
  15: 0.0,
  16: 0.0,
  17: 0.0,
  18: 2.0077499146706285e-05,
  19: 2.0077499146706285e-05,
  20: 2.0077499146706285e-05,
  21: 2.0077499146706285e-05,
  22: 0.0,
  23: 0.0,
  24: 2.0077499146706285e-05,
  25: 0.0,
  26: 0.0,
  27: 0.0,
  28: 0.0,
  29: 0.0,
```