University of New South Wales

COMP9417 Machine Learning & Data Mining

Term 1, 2020

Group Project

Group Member:
Z5241942 Mengqi Deng
Z5238695 Zhifan Zhang

Introduction

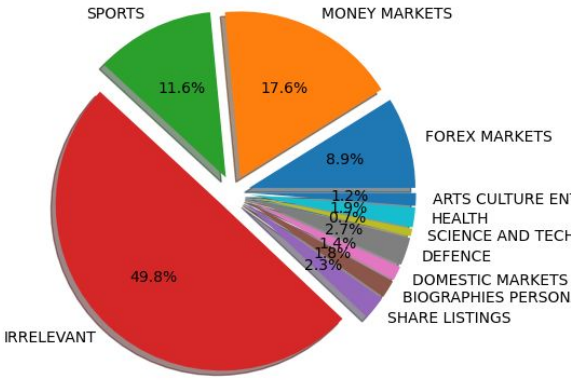
The project is aimed to provide most relevant news from hundreds of latest articles to application users. And in order to offer better service to application users, their preferential topics about articles should also be taken into account.

The main issue can be seen as a topic-based text classification problem. Text Classification problem is the task of classifying a document under predefined topics. This will be achieved with a supervised machine learning classification model that is able to predict the topic of a newly published article and shows the result to its user. The goal of this project is finding the best hyperparameters and best models to offer higher accuracy recommendation. And to achieve this goal, several models will be built and different evaluation metrics will be used to make comparisons and get the best model.

Exploratory Data Analysis

The dataset used in this project is a dataset containing 9500 labeled news articles corresponding to ten topics.

In addition, this dataset is filled with about 48% irrelevant articles (which is marked as *IRRELEVANT*). Because in real life, it is impossible for all articles to be classified into these topics, and users might not be interested in those articles. Up to this point, we see that the class distribution is imbalanced, irrelevant articles take a majority part in our training set. Besides, other topics such as *Science and Technology* articles only occupy a very small part.



Looking into our data, we can get the hundred percent of the observations belonging to each class. And at this moment, we will face the problem caused by highly imbalanced data. This means the model may always be biased towards majority class and the minority class may present poor performance with even 0 prediction accuracy. For the sake of this, we will balance our dataset by oversampling the minority class and undersampling the majority class.

Besides, in the first column of our dataset there are numbers that represent the different articles. The second column is a series of keywords that have been pre-processed from the news articles which means we don't need to do text cleaning to ensure no distortions are introduced to the model.

Methods

feature extraction

As we have talked before, the problem of this project is a classification problem.Then the first step is transforming the text information into numerical data that can be accepted by machine learning models. Because the articles we had are represented by a series of words, so we do not need to do text-cleaning.And we were concerned that a word in a document yields a great part in classification so we decided to choose *TF-IDF* vectors to be our features in Multinomial Naive Bayes model.

TF-IDF is a numerical statistic used to assess the importance of a word in the document or article.Besides, tf–idf value increases proportionally to the number of times a word appears in the document and is offset by the number of documents in the corpus that contain the word, which helps to adjust for the fact that some words appear more frequently in general. This could be computed by this formula:

$$TF - IDF(t,d) = TF(t,d) \times \left\{\log\left(\frac{n}{DF(t)}\right) + 1\right\}$$

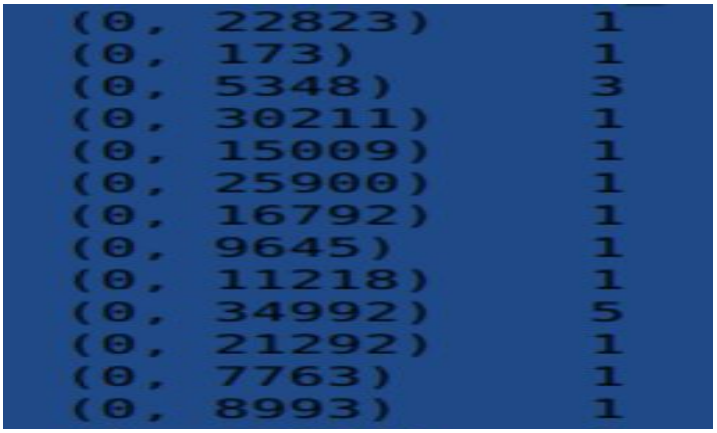
- *TF(t,d):* term frequency (how many times the term t appears in the document d)
- *n:* number of the documents
- *DF(t):* number of documents in the corpus containing the term t

- The effect of adding “1” to the idf in the equation above is that terms with zero idf, i.e., terms that occur in all documents in a training set, will not be entirely ignored.

Machine learning models also require numeric labels to provide a prediction. For this reason, we have to create a dictionary to map each label to numerical ID. So we have created this mapping scheme:

Topic names	Topic ID
IRRELEVANT	0
ARTS CULTURE ENTERTAINMENT	1
BIOGRAPHIES PERSONALITIES PEOPLE	2
DEFENCE	3
DOMESTIC MARKETS	4
FOREX MARKETS	5
HEALTH	6
MONEY MARKETS	7
SCIENCE AND TECHNOLOGY	8
SHARE LISTINGS	9
SPORTS	10

Word count vectorise(only use for decisionTreeClassifier):
The method that has been used is word count vectors, this method defining every column as a set, and each element is the frequency count of each term in each paragraph. It comes from the Word Representation which regarded the word as a vector. The advantage of CountVectorizer is easy to understand and more focusing on each news because word Vectorize will only consider each words’ frequency in each row, which is more related to analyse topics. This method has three processes: pre-processing, tokenizing, n-grams generation. As the assignment says the keyword has been preprocessed and all the words are callable, so, there are no necessaries to add any parameters in the methods. The result after transmitting looks like:



After converting to the array, we can observe that there are a lot of 0 taking place in the array, this is the disadvantage of this method, it takes too much space, resulting in inefficiency.

Predictive Models

We have used two machine learning models to figure out which one may fit better to the data and offer a higher accuracy predictive model.

Multinomial Naive Bayes

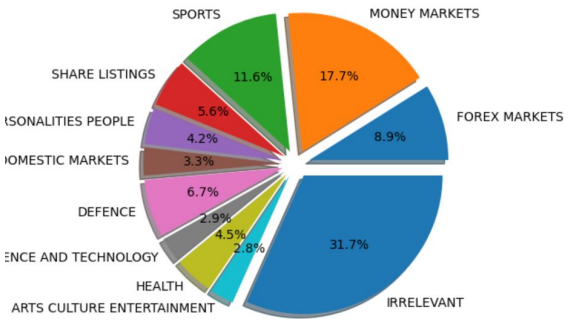
The multinomial model uses vectors of word counts or TF-IDF to represent documents. In this case, it is assumed that each class can be described as a multinomial distribution. Empirical comparison of both models has shown that the multinomial model is usually superior to the Bernoulli model.

Multinomial Naive Bayes implements the naive bayes algorithms for multinomially distributed feature vectors. And the conditional probability of every single feature is computed by formula:

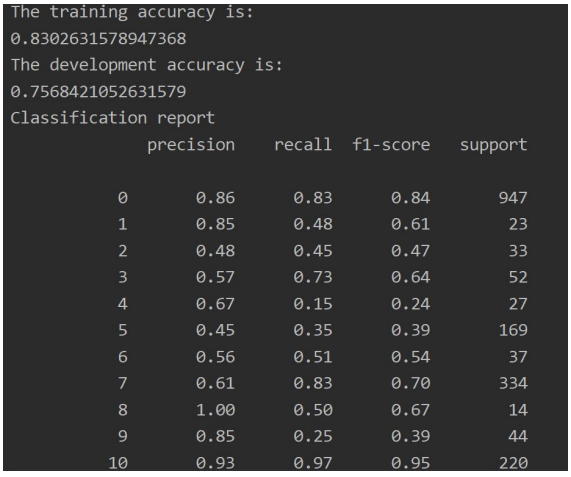
$$p(\mathbf{x} \mid C_k) = \frac{(\sum_i x_i)!}{\prod_i x_i!} \prod_i p_{ki}{}^{x_i}$$

pre-process

As we have talked before, the dataset we have is imbalanced, so we decided to balance the data before we train the model.

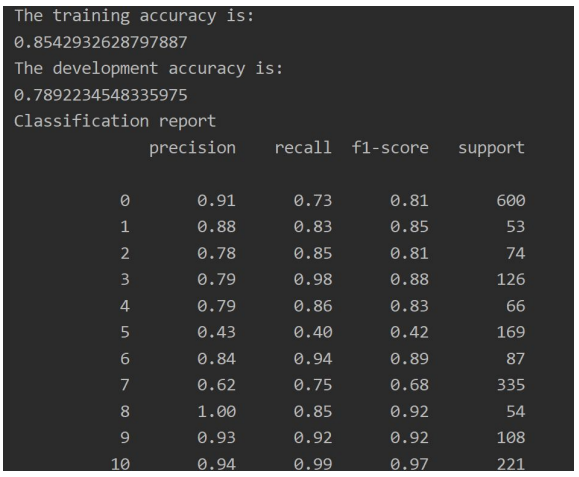


And the results could also show that balancing will help:



```
The training accuracy is:
0.8302631578947368
The development accuracy is:
0.7568421052631579
Classification report
              precision    recall  f1-score   support

           0       0.86      0.83      0.84       947
           1       0.85      0.48      0.61        23
           2       0.48      0.45      0.47        33
           3       0.57      0.73      0.64        52
           4       0.67      0.15      0.24        27
           5       0.45      0.35      0.39       169
           6       0.56      0.51      0.54        37
           7       0.61      0.83      0.70       334
           8       1.00      0.50      0.67        14
           9       0.85      0.25      0.39        44
          10       0.93      0.97      0.95       220
```



```
The training accuracy is:
0.8542932628797887
The development accuracy is:
0.7892234548335975
Classification report
              precision    recall  f1-score   support

           0       0.91      0.73      0.81       600
           1       0.88      0.83      0.85        53
           2       0.78      0.85      0.81        74
           3       0.79      0.98      0.88       126
           4       0.79      0.86      0.83        66
           5       0.43      0.40      0.42       169
           6       0.84      0.94      0.89        87
           7       0.62      0.75      0.68       335
           8       1.00      0.85      0.92        54
           9       0.93      0.92      0.92       108
          10       0.94      0.99      0.97       221
```

(before balancing) (after balancing)

Although the training set accuracy and development set accuracy just go slightly higher, most of the topic classification scores go higher.

Train-Development split

We decided to set apart a development set in order to prove the accuracy when predicting the unseen data. We have randomly chosen 20% data from our dataset (in case some articles may not be obtained).

Feature selection

After extracting features from our articles, we could get that the features may be too many to represent our articles, so we set some threshold to do dimension reduction.

- max_df: We want to figure out if all the words in vocabulary will contribute to the classification. This hyperparameter is to set a threshold when building the vocabulary and ignore terms that have a document frequency strictly higher than the given threshold.
- max_features: This hyperparameter is to set a threshold when building a vocabulary that only considers the top number of the threshold ordered by term frequency across the corpus.
- ngram_range: This hyperparameter is helping to find out whether bigrams will contribute to improve model performance.

- alpha: This hyperparameter is set to imply Laplace smoothing or Lidstone smoothing.

After choosing the above parameters, we use a grid search method to find the best hyperparameter and get the result:

```
{'clf__alpha': 0.01, 'vect__max_df': 1.0, 'vect__max_features': None, 'vect__ngram_range': (1, 2)}
```

And we could get that the model that:

- clf_alpha:0.01 which means the model chose Lidstone smoothing.
- Vect_max_df,Vect_max_deature: which means all the words in vocabulary are used to be the features.
- Vect_Ngram_range: bigrams help to improve our model performance by taking into consideration words that tend to appear together in the documents.

Then we will apply this in our model and we will get the final result.

Decisiontree

Another machine learning method that has been used to classificate topics is being chosen as the decisiontreeclassifier. The decisiontreeclassifier takes the first sample as a node and depends on the feature of the sample, it divides different subsets, making sure all the subsets' values are the same. Then take each subset as the node, use the same process as previously discussed, constructing more nodes. At the end repeat the process until all the features are being finalised. There actually are two requirements for ending. First is finalise all the features. Second, the information entropy increments are small enough. So the information entropy increment is selected as the method of constructing the decision tree.
In order to measure the purity of a subset, the Gini impurity 'G' must be used:

$$G = 1- \sum p * p \text{ (for all the value in the row)}$$

It calculated the probabilities for a row of features and added the value of same features together then minused by 1. Gini coefficients are much faster than Entropy methods since square sum calculations are easier than the logarithm, but their ideas are the same.

Tree implementing

The tree models are very likely to have overfitting, escipally at the case where features are far more than labels, or any case that the training sample is few. The pruning is when the information entropy cannot decrease further then it will stop constructing splitted nodes. Therefore, in order to avoid overfitting.

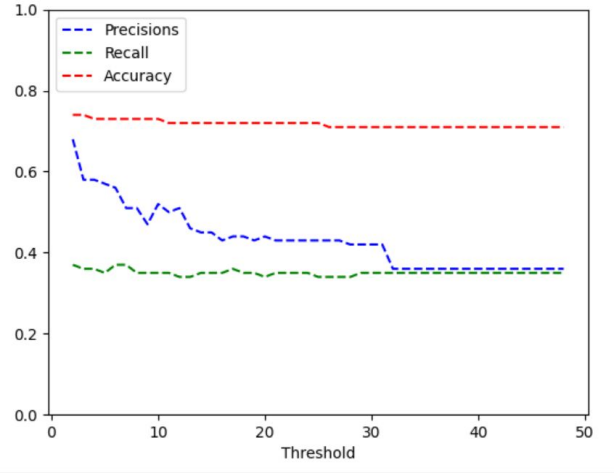
Tree pruning

It is needed to prune the tree, there are two ways of pruning, pre-pruning and post-pruning. Pre-pruning is pruning the tree before the decision tree is constructed setting a threshold, if the information entropy is less than this threshold, we can not only keep decreasing the entropy but also stop constructing the splitted node. Post-pruning is pruning the tree after the model is constructed. It deletes some subtree and then uses the root of subtree as this new leaf node, next is regarding the class that has the most number of samples as the class of leaf node.

Find the best parameter

The pruning method can be applied as changing the parameters. One of the most influenced parameters is max_depth, if the input data set has a tremendous amount of features and labels then the limit on maximum 'max_depth'is needed for pruning. What's more, the 'min_impurity_decrease' is also very important. This value limited the decision tree growing. In order to develop these two parameters.

Minimum sample leaf: The minimum number of samples required to be at a leaf node, achieved by the iteration from the integer 2 to 50.



Min_sample_leaf development procedure

This procedure is achieved by iterating from number 1 to 50, since the more data set the higher reliability it will be. In observation if the value of Min_sample_leaf is too high then in evaluation stage the result looks as:

	precision	recall	f1-score	support
0	0.57	0.95	0.71	202
1	0.00	0.00	0.00	34
10	0.49	0.45	0.47	66
2	0.00	0.00	0.00	7
3	0.00	0.00	0.00	9
4	0.00	0.00	0.00	10
5	0.00	0.00	0.00	6
6	0.00	0.00	0.00	6
7	0.00	0.00	0.00	1
8	0.00	0.00	0.00	6
9	0.00	0.00	0.00	53
accuracy			0.56	400
macro avg	0.10	0.13	0.11	400
weighted avg	0.37	0.56	0.44	400

The low recall value means it found less true positive result in the true set. Data topics with number ‘4,5’ are being pruned out of the tree which is overfitted, so the number of iteration has been reduced to 5, it makes sure that the majority data can be covered. The result are follows:

	precision	recall	f1-score	support
0	0.75	0.98	0.85	202
1	0.77	0.50	0.61	34
10	0.88	0.76	0.81	66
2	0.50	0.29	0.36	7
3	0.75	0.33	0.46	9
4	0.50	0.20	0.29	10
5	0.00	0.00	0.00	6
6	0.75	0.50	0.60	6
7	0.00	0.00	0.00	1
8	0.00	0.00	0.00	6
9	0.95	0.75	0.84	53
accuracy			0.79	400
macro avg	0.53	0.39	0.44	400
weighted avg	0.76	0.79	0.76	400

Weight adding

We will use a function to estimate class weights for unbalanced datasets and automatically rearrange data and calculate the weight for each topic that appears in the set. In order to get higher accuracy we will tune these three parameter:

- class_weight:which should be ‘balanced’, the class weight will given by n_samples / (n_classes * np.bincount(y));
- the classes is the array frame of the classes given by the data;
- y: the data of labels input.

For example, if the input value is 9000 pairs train data, then the result is:

```
{'0': 0.1811139281258803, '1': 1.041059449456396, '2': 7.438842975206612, '3': 5.178941311852705, '4': 3.2994868035190614, '5': 6.494227994227995, '6': 4.649276859504132, '7': 11.859025032938076, '8': 3.859777015437393, '9': 0.7830361026533276, '10': 0.5268980858163086}
```

In the model, parameter ‘class_weight’ is equal to this result.

As introduced, decisiontreeclassifier were once computed with weight adding, however the result is horrible, due to the overfitting issue, then another method is used which implements class weight by GridSearchCV, the result is that programmer had to abundant after 40 minutes no responding.

GridSearchCV

We will use a function to map these two parameters, depending on the results of best accuracy score then it returns the best_params.

- *Estimator* is the model that we selected,
- *param_grid* is a diction that combines the max_depth and the min_impurity_decrease,
- *cv* determines the cross validation splitting strategies,
- *verbose* is indicating control of verbose.

Train-Development split

There are two stages of the developing the first time is the prime model which have no parameter changing only one decision tree classifier implemented it performed not very well but it is expected, accuracy with train data and test data:

Training data: the accuracy score for training set is 0.99
Testing data: the accuracy score for test set is 0.69

The accuracy is too great which is overfitted, the node are expanding without any limitations.Then after the implementation of using the min_sample_leaf, max_depth, min_impurity

Training data: the accuracy score for training set is 0.73
Testing data: the accuracy score for test set is 0.65

The data becomes more reliable since it will no longer be bothered by the accuracy paradox and less overfitting, because adding restriction pruning processes resolved the issue of overfitting.

The grid parameter for training set and test set are acceptable, with very low min_impurity and relatively lower max_depth

```
{'max_depth': 10, 'min_impurity_decrease': 0.0}
```

```
{'max_depth': 9, 'min_impurity_decrease': 0.0}
```

Results

The diagram follows the screenshot of the classification report, the number on the left column represents topics of news correspondingly: **0.IRRELEVANT 1.ARTS CULTURE**

ENTERTAINMENT 2. BIOGRAPHIES PERSONALITIES PEOPLE 3. DEFENCE 4. DOMESTIC MARKETS 5. FOREX MARKETS 6. HEALTH 7. MONEY MARKETS 8. SCIENCE AND TECHNOLOGY 9. SHARE LISTINGS 10. SPORTS.

- Cross-validation results:

```
Accuracy =  0.8237375040345594
```

(Multinomial Naive Bayes)

```
Accuracy =  0.6868128200382503
```

(DecisiontreeClassifier)

- Final results for each class calculated on the whole test set using the final selected method with its hyper-parameters：

	precision	recall	f1-score
0	0.84	0.88	0.86
1	0.25	0.33	0.29
2	1.00	0.13	0.24
3	1.00	0.31	0.47
4	0.00	0.00	0.00
5	0.47	0.44	0.45
6	0.86	0.43	0.57
7	0.52	0.72	0.61
8	0.00	0.00	0.00
9	1.00	0.29	0.44
10	0.95	0.95	0.95

(Multinomial Naive Bayes)

	precision	recall	f1-score
0	0.77	0.86	0.81
1	0.58	0.45	0.50
10	0.65	0.67	0.66
2	0.53	0.27	0.36
3	0.00	0.00	0.00
4	0.64	0.33	0.43
5	0.60	0.60	0.60
6	0.47	0.34	0.39
7	0.00	0.00	0.00
8	0.58	0.54	0.56
9	0.82	0.87	0.85

(Decision Tree Classifier)

- Final article recommendations using the final selected method with its hyper-parameters:

```
[1, [9703, 9834, 9952]]
[2, [9940, 9933, 9878, 9758, 9575, 9842, 9654, 9546, 9644, 9981]]
[3, [9559, 9773, 9576, 9842, 9670, 9783, 9607, 9616, 9777, 9668]]
[4, [9796, 9994]]
[5, [9588, 9671, 9823, 9632, 9682, 9851, 9693, 9696, 9625, 9704]]
[6, [9873, 9621, 9661, 9929, 9807, 9947, 9617, 9978, 9982, 9575]]
[7, [9853, 9829, 9835, 9606, 9845, 9602, 9855, 9594, 9860, 9863]]
[8, [9604, 9722, 9929]]
[9, [9518, 9562, 9601, 9654, 9667, 9972, 9999]]
[10, [9657, 9513, 9800, 9964, 9514, 9582, 9809, 9813, 9580, 9817]]
```

（Multinomial Naive Bayes, final selected model）

(The given data follows should plus 9501 is the article topic number, due to the issue that the article number column is dropped for easier mupliting)

```
0 :
dict_keys([296, 356, 13, 253, 457, 0, 1, 4, 6, 8])
1 :
dict_keys([29, 76, 164, 178, 391, 2, 83, 203, 217, 350])
2 :
dict_keys([202, 333, 450])
3 :
dict_keys([25, 282, 288, 431, 438, 80, 376, 481, 329, 144])
4 :
dict_keys([58, 205, 212, 220, 238, 272, 340, 75, 269, 111])
5 :
dict_keys([295, 492])
6 :
dict_keys([145, 306, 445, 120, 160, 234, 309, 371, 409, 480])
7 :
dict_keys([103, 221, 427])
8 :
dict_keys([17, 61, 100, 153, 166, 470, 497])
9 :
dict_keys([7, 12, 40, 67, 68, 73, 81, 95, 119, 155])
10 :
dict_keys([93, 210, 228, 275, 322, 343, 461, 47, 124, 239])
```

（Decision tree classifier)

Discussion

precision It is calculated by the true positive ／retrieved set. It will count the total positive case that model measured, then the higher true positive cases there are the better precision it will be. It can be treated as the confidence score of the model. **Multinomial Naive Bayes have a better precision score.**

The result of Multinomial Naive Bayes is definitely better than decisiontreeclassifier. Since there are significantly amount of data set fed to model, however the decisiontreeclassifier is a slow algorithm and took nearly 3 minutes to finish calculating the best parameters `2.9min finished`

F1-score: the Multinomial Naive Bayes has better F1-score, F1-score combines precision and recall, the calculation is 2*precision*recall / (precision + recall) . Classifierdecisiontree

model is trying to be very conservative then the precision will be high and recall will be low. If it is trying to cover more samples, then it will likely make mistakes that result in a high recall but low precision. It is calculated by true positive ／ golden se. For example the test set Topic6 has 100 positive cases, if the model could fit 47 positive cases then the recall is 47%. So that it could be more reliable.

For all the data input, there are multi-class samples with more than 10 labels then the marco-average will be better performer than micro-average. Since macro will be influenced by minority class, escapilly signifiate on small classes.

Conclusion

In the ending stage of comparing the data, The Multinomial Naive Bayes is better than decisiontreeclassifier on this task. The precision, recall and f1-score are slightly better, what's more the run time is much faster than decisiontreeclassifier. Nevertheless, the TF-IDF algorithm performs better than wordCountVectorise, since wordCount method will only transmit the words array into the confusion matrix and a lot of space is wasted, it will perform extremely poorly compared to the TF-IDF in handling large data set cases. The new knowledge that is being learned is the word converting, which is important to further study.

Reference

[1] scikit-learn developer, 007 - 2019, [Blog] sklearn.model_selection.GridSearchCV, Available at https://scikit-learn.org/stable/modules/generated/sklearn.model_selection.GridSearchCV.html/ [Accessed at 25 April.2020]

[2] scikit-learn developer, 007 - 2019, [Blog] sklearn.utils.class_weight.compute_class_weight(*class_weight*, *classes*, *y*), Available at https://scikit-learn.org/stable/modules/generated/sklearn.utils.class_weight.compute_class_weight.html [Accessed at 25 April.2020]

[3] CristianPadurariu, Mihaela ElenaBreaban, 2019, *Post title*. Dr,[Blog] Dealing with Data Imbalance in Text Classification, Available at https://www.sciencedirect.com/science/article/pii/S1877050919314152#cebibsec1 [Accessed at 22 April.2020]

[4] Hermann Ney, January 2002, *Post title. Dr,* [Blog] Reversing and Smoothing the Multinomial Naive Bayes Text Classifier, Available at https://www.researchgate.net/publication/221383148_Reversing_and_Smoothing_the_Multinomial_Naive_Bayes_Text_Classifer [Accessed at 23 April.2020]