Classification of Categories in News articles

Data Mining – II, Final Report

1. Task Description

In This Project, we are scraping data from google news api and that includes the news titles and their respective categories. Our final dataset consists of 4722 entries which consist of news articles from all the categories.

Here, we are using classification to predict the data. In this project, we had used many types of Classifiers like Random Forest classifier, K-Nearest Neighbor classifier, logistic regression Classifier, Gaussian Naïve Bayes Classifier to predict the Accuracy, Classification Report. By Running all the Classification methods, we will get different results, and these are discussed in below sections.

2. Dataset Description

The dataset is fetched using google api on daily basis and collected data for 19 days starting from 12 March to 30 March. We collected and stored data from scratch, which in total came out with 4722 number of articles from our 6 pre-defined categories

- Business
- Technology
- Health
- Science
- Sports
- Entertainment

Our final dataset consists of two columns, first is column of news titles and second one consists of categories of that news articles which is also collected from the same api.

This dataset will be used for training our model which can help our model in learning process and help in finding patterns for the classification of news items.

3. Description of Methods

In any data mining project, the data is the most important starting point. After scraping the data, we need to load the data into the jupyter notebook. After loading the data, we need to perform the analysis of the data like checking the stats such as news lengths, number of news in all categories or individually as well.

The next step is the Data Cleaning part, where we cleaned our string data. In cleaning of string data, we cleaned all the punctuation marks, special characters, converting all the words in lower case words, removing possessive nouns, and finally moving towards the NLP tasks of stemming and lemmatization and removal of stop words.

If we must define Stemming, Stemming is the method of reducing inflected (and sometimes derived) phrases to their word stem or base-generally a written word form-in linguistic morphological and information retrieval. The stem does not have to be the same as the word's morphological root; it's

typically enough that terms mapped to the same stems, even if that stem isn't a valid root in and of itself.

Strings like 'cats', 'cat-like', and 'catty' should be identified by an English stemmer acting on the stem 'cat'. ' Fishing', 'fishery', and 'fisher' might all be reduced to the stem fish using a stemming algorithm. The stem does not have to be a word; for example, the Porter method minimizes the stem 'argu' to 'argue', 'argued', 'argues', 'arguing', and 'argus'.

Next Step of Data processing includes labelling our categories and our String news articles into numbers for better understanding of our models.

To convert categories into different labels we manually did the same by providing an integer value to each kind of categories. And for converting our string data into numerical format we used TF-IDF method for the conversion

If we talk about the working of TF-IDF, the text vectorizer term frequency-inverse document frequency converts the text into a useable vector.

Term Frequency (TF) and Document Frequency (DF) are combined in this idea (DF).

The term frequency refers to the number of times a phrase appears in a document. The frequency of a phrase in a text reflects how essential it is. Term frequency displays each text in the data as a matrix with the number of documents in the rows and the

number of different words in the columns.

The weight of a term is determined by the inverse document frequency (IDF), which seeks to minimize the weight of a phrase if the term's occurrences are dispersed throughout all documents.

After all the conversion our data is now ready to be used for classification process.

3.1 Classification process

Classification is the process of creating a model which could give us the predictions for future entries based on our old records.

Classification consists of two steps:

1. Building a classifier: first we build a classifier based on our records with class labels, then classifier builds up the models based on patterns available in the training data we provided.

2. Using classifier for classification: Now we use that classifier on test data to check for the accuracy of predication given by classifier.

After successful testing and validation of classifier, we can use it for our future predictions.[1]

We used 4 classification methods:

- Random Forest: A decision tree is a structure that includes a root node, branches, and leaf nodes. Each internal node denotes a test on an attribute, each branch denotes the outcome of a test, and each leaf node holds a class label. The topmost node in the tree is the root node. random forest is simply a collection of decision trees

whose results are aggregated into one final result.

- Logistic regression: Logistic regression is a classification algorithm, used when the value of the target variable is categorical in nature. Logistic regression is most used when the data in question has binary output, so when it belongs to one class or another, or is either a 0 or 1.
- K-means classification: K-means classification is a clustering algorithm which divides observations into *k* clusters. Since we can dictate the number of clusters, it can be easily used in classification where we divide data into clusters which can be equal to or more than the number of classes.
- Naïve Bayes Classification: It is a classification technique based on probability Bayes' theorem with an assumption of independence among predictors. In simple terms, a Naive Bayes classifier assumes that the presence of a particular feature in a class is unrelated to the presence of any other feature in the class.

Steps for implementation of these classifier in our project:

1. First, we will try these classifiers in base form.

2. Secondly, we'll define the metric we'll get when measuring the performance of a model. In this case we'll use the accuracy.

3. After seeing the best fit model. we will improve the classifier using parameter tuning.

4. We'll perform a Randomized Search Cross Valid ation process in order to find the hyperparameter re gion in which we get higher values of accuracy.

5. Once we find that region, we'll use Grid Search Cross Validation process to exhaustively find the best combination of hyperparameters.

6. Once we obtain the best combination of hyperpa rameters, we'll obtain the accuracy on the training data and the test data, the classification report and the confusion matrix.

7. Finally, we'll calculate the accuracy of a model with default hyperparameters, to see if we have achieved better results by hyperparameter tuning.

8. Whichever model have the highest accuracy we are going to use that model as our final model.

4. Results

After applying all the classifier in our training dataset, we get accuracies like this:

```
Training accuracy for Random Forest
0.9908889436099483
Training accuracy for K Neighbors
0.7365180989903964
Training accuracy for linear regression
0.8468357547402118
Training accuracy for naive bayes
0.6067471066239842
```

In this we can clearly see the highest accuracy was given by Random Forest classifier so we will apply the hyperparameter tuning and will try to improve the accuracy of our model and compare the results.

Hyper parameter is multiple trials that are run in a single learning job to perform hyperparameter optimization. Each test is a complete run of your learning application with parameters for your selected hyperparameters set within the limitations you define. The APT training service maintains track of each trial's findings and adjusts for future trials. When the work is completed, you will receive a report of all the tests as well as the most effective value configuration based on the criteria you select. Therefore, for our Random Forest, these are all the available parameters which we can define for tuning,

```
Parameters currently in use:
{'bootstrap': True,
'ccp_alpha': 0.0,
'class_weight': None,
'criterion': 'gini',
'max_depth': None,
'max_features': 'auto',
'max_leaf_nodes': None,
'max_samples': None,
'min_impurity_decrease': 0.0,
'min_samples_leaf': 1,
'min_samples_split': 2,
'min_weight_fraction_leaf': 0.0,
'n_estimators': 100,
'n_jobs': None,
'oob_score': False,
'random_state': 8,
'verbose': 0,
'warm_start': False}
```

parameter we chose for Randomized search

```
{'bootstrap': [True, False],
'max_features': ['auto', 'sqrt'],
'min_samples_leaf': [1, 2, 4],
'min_samples_split': [1, 2, 4, 6],
'n_estimators': [50, 87, 125, 162, 200],
'random_state': [5, 28, 52, 76, 100]}
```

Parameter we chose for Grid search are

```
 {'bootstrap': bootstrap,
 'max_features': max_features,
 'min_samples_leaf': min_samples_leaf,
 'min_samples_split': min_samples_split,
 'n_estimators': n_estimators,
 'random_state':random_state}
```

We used two tuning methods first is RandomSearchCV and Second one is GridSearchCV and as far main difference in both that in grid search, we define the possibilities and train the model and it start implementing them in ordered manner, whereas in RandomizedSearchCV, the model chooses the combinations at random.

Results of accuracy from RandomSearchCV

```
The best hyperparameters from Randomized
Searches are:
{'random_state': 52,
 'n_estimators': 200,
 'min_samples_split': 6,
 'min_samples_leaf   ': 1,
 'max_features': 'auto',
 'bootstrap': False}

The mean accuracy of a model with these
hyperparameters is: 0.824426307241453
```

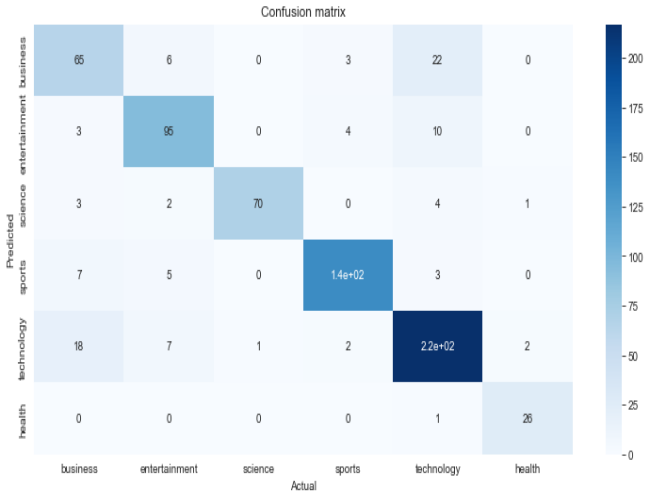
Results from GridSearchCV

```
The best hyperparameters from Grid Search
are:
{'bootstrap': False,
 'max_features': 'auto',
 'min_samples_leaf': 1,
 'min_samples_split': 4,
 'n_estimators': 200,
 'random_state': 52}

The mean accuracy of a model with these
hyperparameters is: 0.8409184091840919
```

From Both the methods, we can conclude that accuracy for Random Forest in its base form is the highest with 99.08%, therefore we will choose our final model as baseline model. Upon further checking the model on our test data we got accuracy of 85.50 %. We have observed the difference of training and testing accuracy, probably the words in

the testing set will never appear in the training set, so model doesn't know how to classify them properly. As for the results of individual categories is, we will examine the confusion matrix.



5. Conclusion

By performing the above models, we get the different accuracy score values based on the different validations. From this, we concluded that the Random forest in base form is the best fit for this project because it is having the highest accuracy when compared with others. We can see that it is misclassified the business and technology compare the others category. As technology has important effect on business. Business and technology are entangle, there is higher chance of miscategorized.

6. Future Work

I will be looking to improve our model using deep learning concepts and will try have better classification accuracy using BERT models also.

7. References

1. Top 6 Machine Learning Algorithms for Classification.

https://towardsdatascience.com/top-machine-learning-algorithms-for-classification-2197870ff501

2. Overview of hyperparameter tuning | AI Platform Training | Google Cloud.

https://cloud.google.com/ai-platform/training/docs/hyperparameter-tuning-overview

3. Hyperparameter tuning for machine learning model.

https://towardsdatascience.com/hyperparameter-tuning-for-machine-learning-models-1b80d783b946