FIT5149 APPLIED DATA ANALYSIS

Assignment 02 – Authorship Profiling

Group 02

Group member 1: Raguram Ramakrishnasamy Dhandapani
Student ID: 30151325

Group member 2: Thirugnanam Ramanathan
Student ID: 30404975

Group member 3: Nuwan Chamila Withana Gamage
Student ID: 29255066

Contents

1. Task and Dataset Description

1.1 Task Description

Authorship analysis deals with the classification of texts into classes based on the stylistic choices of their authors. The assignment focuses on Gender classification on Authorship profiling, where based on the text content (from any post/blog), the Gender of the author needs be identified.

The data containing Twitter post of 3600 authors and respective Gender is provided where each author would have multiple posts. The task for this assignment is to develop a classifier that can find the Gender of the author given a set of twitter texts.

1.2 Dataset

Two datasets are provided – one for training and other for testing the classifier.

Both the datasets contain the ID and Gender of the different authors. Each author has a set of twitter texts that is provided in a separate XML file.

Based on the IDs, the corresponding text for training and testing is read and processed from each of the XML files provided (one file per author).

2. Text Preprocessing and Feature Generation

The first task of the project is to extract the texts corresponding to each author and prepare a corpus for feature set generation.

2.1 Extracting texts

Section in Code: **Extracting data from XML files**

Each author is provided with an XML file that contains a set of twitter posts made by them. Since the text is in the XML format, **Regex** is used to extract the text from each file, where redundant and unnecessary content are removed. For example, the tags governing each twitter post (<*document*>), the text at the start of each twitter post (*'![CDATA['*) are removed.

The extracted text from each XML file is stored in a list that acts as a final corpus for feature generation. This contains text which would be processed further using Text processing algorithms.

2.2 Text Processing

Section in Code: **Text preprocessing**

Different set of preprocessing measures are followed in this process. They are:

(a) Case Normalisation

The Text is converted to lower case since the upper-case and lower-case representation of the same word would be considered as different features in Text processing. Hence all the contents are converted to lower case representation

(b) Tokenisation

The text cannot be processed as a whole corpus. It should be broken down into smaller components called tokens. There are different approaches for creating the tokens. In this project Regex is used to create the tokens.

The Regex syntax of **[a-zA-Z]+** is used where the **only text** are considered. All the words with length of 2 or more are considered.

(c) Document Frequency

The number of documents a word is present in is called the document frequency of a word. In this analysis, **the minimum document frequency is fixed at 5%** and **maximum frequency is 95%** of documents. This is kind of standard to consider important words that occur in few documents and removing the more frequently occurring words.

(d) Stop words

The words that have the least lexical meaning to the context and the most frequently occurring words in the document are considered the Stop words. These words are removed from the analysis as they do not contribute as key features.

The English stop words from the NLTK package and punctuations are set as the stop words for this analysis.

(e) Lemmatisation

Lemmatisation in linguistics is the process of grouping together the inflected forms of a word so they can be analysed as a single item, identified by the word's lemma, or dictionary form. It takes into consideration the morphological analysis of the words.

The Lemmatisation is done using WordNetLemmatizer where each word in the document is lemmatized.

(f) N-gram Features

Considering Unigrams, Bigrams and Trigrams for the analysis would improve the analysis since we consider different set of features and each of them would add a specific value for the feature set

2.3 TF-IDF Vectorizer

Final implementation of above-mentioned steps is done using the TF-IDF vectorizer in Python's SKLEARN package, where each word in the corpus is assigned to a TF-IDF weight which is used in the model building and analysis.

TF-IDF stands for 'Term Frequency — Inverse Document Frequency' which is a technique to quantify a word in documents, and generally compute a weight to each word which signifies the importance of the word in the document and corpus.

```
class token_stem(object):
    def __init__(self):
        self.wnl = PorterStemmer()
    def __call__(self,doc):
        return [self.wnl.stem(t) for t in RegexpTokenizer('[^\d\W]{4,}').tokenize(doc)]
```

```
vectorizer=TfidfVectorizer(analyzer='word',input='content',
                           lowercase=True, ngram_range=(1,3),
                           min_df=0.02,max_df=0.8,
                           stop_words=set(list(string.punctuation) + stopwords.words('english')),
                           tokenizer = token_stem())
```

The above snippet of the code creates the Vectorizer that is used to convert the corpus of text into TFIDF matrix, while applying all the mentioned pre-processing steps.

The main objective of the vectorizer is that it is used to fit the Train data to create the set of features and transform the Test set using the same function so that the feature set remain the same.

2.4 Final dataset features

The dataset contains the TF-IDF sparse matrix as the Input and the corresponding labels as Output.

Training Dataset:

Input Matrix dimension – Based on the parameter settings of TF-IDF Vectoriser the dimension is 3100*1865

Output Dimension – There is only one output value with a dimension of 3100*1 (with 0/1)

Test Dataset:

The test dataset is transformed using the same vectorizer with a dimension of 500*1865 for input data used to predict the labels (500*1)

After building the Vectorizer and creating the TF-IDF sparse matrix, the next step is to build the classification models.

3. Model Selection

Section in Code: **Model Building**

The first step in Model building is to choose the algorithms that can be used for the analysis so that the objective of the task is achieved. The task in this project is to correctly predict the correct label given the set of twitter texts and get **maximum accuracy out of the test set** for the classification.

3.1 Model Definition

Based on the data provided, the labels are of Male and Female and hence it is **a Binary classification problem**. So, the most common algorithms that can be used for this task are

- Logistic Regression

Logistic regression is a statistical model that in its basic form uses a logistic function (Sigmoid Function) to model a binary dependent variable. The sigmoid function classifies the variable as 1/0 based on the sigmoid graph with the limit as 0.5. The predictions greater than equal to 0.5 are considered 1 and less than 0.5 are considered 0.

- Linear SVC

The objective of an SVC (Support Vector Classifier) is to fit to the data provided, returning a "best fit" hyperplane that divides or categorizes the data into different planes based on labels. It is more suited for Binary classification problems but also works for multi label classification. SVC here uses the Linear kernel.

- Random Forest

Random forest is an ensemble technique of Machine Learning, where many individual decision trees operate as a single entity. Each individual tree in the random forest spits out a class prediction and the class with the most votes becomes our model's prediction. Many relatively uncorrelated models (trees) operating as a committee will outperform any of the individual constituent models.

- K Nearest Neighbours

The KNN algorithm assumes that similar things exist in nearby proximity. K-nearest neighbours (KNN) algorithm uses 'feature similarity' to predict the values of new datapoints which further means that the new data point will be assigned a value based on how closely it matches the points in the training set.

- Decision Tree

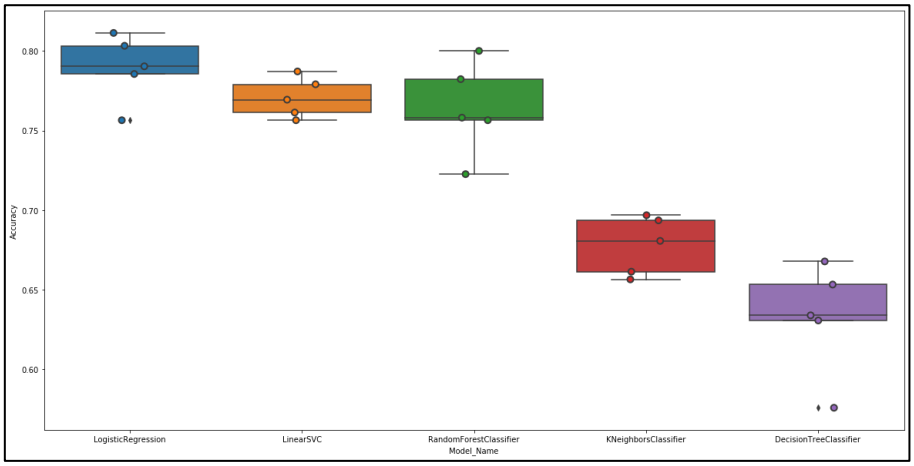
The goal of using a Decision Tree is to create a model that can be used to predict the class of the target variable by learning simple decision rules inferred from training data. In Decision Trees, for predicting

a class label for a record we start from the root of the tree and then compare the values of the root attribute with the record's attribute. Based on comparison, we follow the branch corresponding to that value and jump to the next node and finally arrive at the class label.

3.2 Cross Validation

For all the above-mentioned algorithms, the train data is fit to see which model best fits the data. A cross validation is built on 5-fold with key metric as Accuracy and below plot shows the results of the cross validation of the 5 algorithms.

Based on the above analysis, **Logistic Regression, Linear SVC and Random Forest** works better on the train data.



Next step in this process is to fine tune the above 3 models to get the best accuracy on the test set prediction.

4. Fine tuning Models

4.1 Approach Followed

In the process of Fine-Tuning the models, the approach followed is for the entire set of train features, the Hyper parameters of each of the models are tuned for different values and the best model is found using **Grid Search Cross Validation** where **Accuracy** is the deciding factor and 5-fold cross validation.

In this analysis below steps have been followed for each of the algorithm:

4.2 Tuning Logistic Regression

The Logistic Regression tuning was done by adjusting the key hyper parameters such as **the C value** and the **solver**. On analysing over the C-value as 0.01, 0.1 and 1 to 1e3 in space of 50 intervals across each of the different solvers present (newton-cg, lbfgs, liblinear, sag, saga), the best combination was found to be **C=1.52642** and **solver is newton-cg.** The accuracy obtained on test set was **81.2%**

4.3 Tuning Linear SVC

Linear SVC was tuned by adjusting the key hyper parameters which is **the C value**. On analysing over the C-value from 0.01 to 10 with irregular intervals, the best accuracy was **81.0%** found at **C=0.13.**

4.4 Tuning Random Forest

Random Forest was tuned by adjusting the key hyper parameters such as n_estimators and max_depth. The n_estimators is set for different values of 100 to 500 (in terms of 100) and max_depth was set for 7 to 12 intervals of 1with the criterion as **'gini'** and max_features as **auto.** The accuracy obtained for the best parameters of n_estimator of 200 and max_depth of 10 was **75.8%.**

4.5 Final model

Comparison on all the above models based on the accuracy led to finalising the best model.

Based on the test accuracy obtained, Logistic regression best fits the data with **C** value of **1.52642** and **solver** as **newton-cg** with the test accuracy of **81.2%.**

5. Conclusion

The study is performed on the Gender classification on the authors of the twitter texts involved extracting the data, pre-processing them and finally embedding the text using TF-IDF. After the embedding process, different classifiers were tested on the data find out the best classifiers and the top 3 classifiers were fine-tuned for different hyperparameters to find the best test accuracy.

The final classifier built works well for this dataset but cannot be generalised for other datasets and other types of textual analysis such as Language classifier, Sentiment analysis etc.

More complex algorithms can be built using Neural nets and Deep learning approaches that work well with the textual data analysis such as Convolutional Neural Nets, LSTM models and many more.

6. References

https://towardsdatascience.com/tf-idf-for-document-ranking-from-scratch-in-python-on-real-world-dataset-796d339a4089

https://pythonprogramming.net/linear-svc-example-scikit-learn-svm-python/

https://towardsdatascience.com/understanding-random-forest-58381e0602d2

https://www.tutorialspoint.com/machine_learning_with_python/machine_learning_with_python_knn_algorithm_finding_nearest_neighbors.htm

https://www.kdnuggets.com/2020/01/decision-tree-algorithm-explained.html