

Big Data Mining Techniques assignment report

Fall semester 2023-2024

Dimitrios Tsiompikas

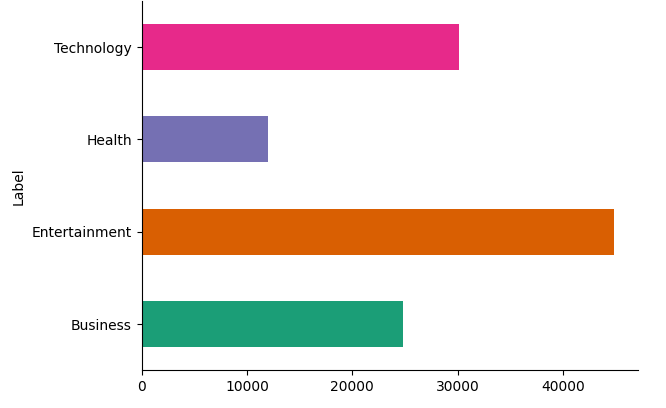
AM: 7115112300036

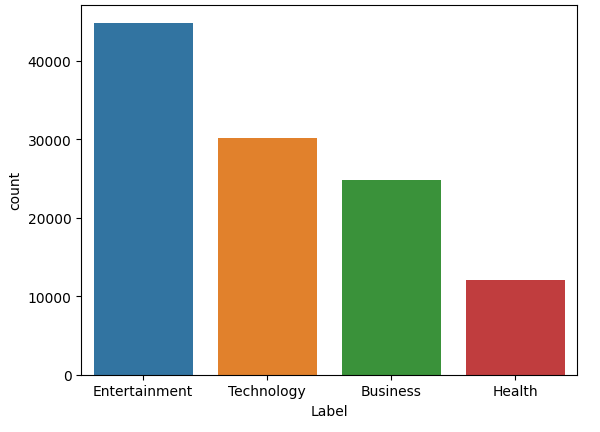
**Part 1: Text classification**

Initially, I loaded the data by using the Pandas library and I began cleaning the dataset from any NaN values. The dataset did not contain any , and in general it was a very easy dataset in terms of interpretation from the known python libraries that are used for NLP purposes. Afterwards, I started the data pre-processing part by doing the following: I used several libraries including NLTK and seaborn for plotting. I also used the following techniques: I lowered all text in the articles, removed all special characters, removed all stopwords (using NLTK’s stopword set and some of my own that I noticed were very much used in the dataset without providing any meaningful context) and finally I lemmatized the Content column in order to maximize results during model training.

**Question 1.1**

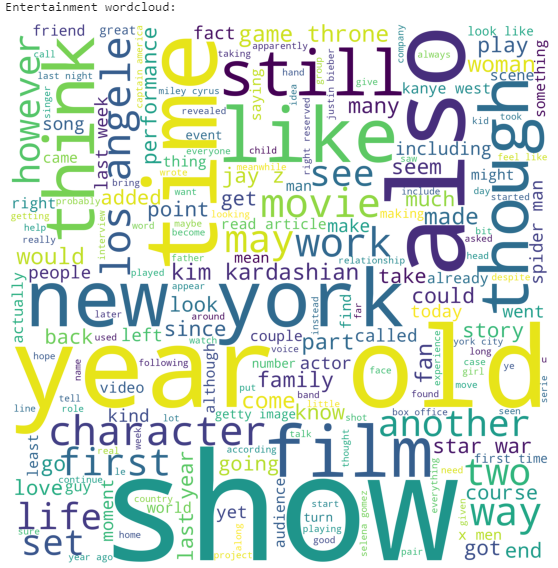
Below you will find some diagrams regarding the dataset and the wordclouds:





Here , in the distribution diagram and countplot we can see the distribution of data and how many articles each label has. Entertainment has the most articles as a label and health has the least, which could seem like a problem due to the uneven distribution of articles but later on the results will show that this is false.

Entertainment word cloud:



Here we can see many words regarding tv shows, films , characters of films etc and some music related words which contain names like Miley Cyrus and Jay-Z.

Business word cloud:



Here we can see business-related words, like company, percent , central bank, market share, per share etc. which help with business articles being labeled correctly.

Technology word cloud:



Here we see names of Technological “Giants”, companies that have made major advancements in the field of Computer Science and Computer Engineering like Google, Apple, Microsoft , Facebook and Samsung. Also some other words like operating system , device, hardware etc that relate to the Technology label.

Health word cloud:



Here we see many words regarding health like patient, disease control, doctor, public health, treatment, hospital, mental health etc, which give a pretty good understanding of which article relates to health.

**Question 1.2**

In this part I made several tests with scikit-learn algorithms such as XGBoost, SVM algorithms, Random Forest and Logistic Regression to predict the labels of the test set. Firstly, I used the count vectorizer on the Content column of the train set by scikit-learn to create a Bag of Words in order to run SVM and Random Forest with BOW and also used the TruncatedSVD algorithm to create the SVD technique for SVM and Random Forest with SVD. BoW had better performance than SVD.

For SVM, I used the LinearSVC method because it was the fastest implementation of the model taking only 30 seconds to run for 1000 iterations (both for SVD and BOW). For Random Forest , I used 20 estimators (trees) because it was really slow , taking an average of 10 minutes per run , quite costly in our case where we want to run multiple experiments to optimize the algorithm (for BOW) and 100 estimators for SVD since it was faster there taking only 5 minutes to complete per run (more estimators did not improve performance of the model however, hence why I stayed with 100).

In order to beat the benchmark, I used the hashing vectorizer and combined the title and content columns to create the combined column which was used to bring the best predictions possible. The model that was used was the LinearSVC once again since it provided the best results out of all tests.

Below you will find the table with the results of my experiments:

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| **Statistic Measure** | **SVM (BOW)** | **Random Forest (BOW)** | **SVM (SVD)** | **Random Forest (SVD)** | **XGBoost**  **(hash vec)** | **Logistic**  **Regression**  **(hash vec)** | **My Method**  **(LinearSVC with hash vec)** |
| Accuracy | 96% | 92% | 90% | 91% | 81% | 96% | 97% |
| Precision | 96% | 92% | 90% | 91% | 81% | 96% | 97% |
| Recall | 96% | 92% | 90% | 91% | 81% | 96% | 97% |

Some final comments are that XGBoost was pretty slow (just like RF , they’re ensemble algorithms after all) hence the need for low number of estimators (20) and low depth (3). Logistic Regression was really good for this task (hit a pretty good 96%, however LinearSVC surpassed it).