MACHINE LEARNING PROJECT (COMP9417)

SENTIMENT ANALYSIS ON MOVIE REVIEWS

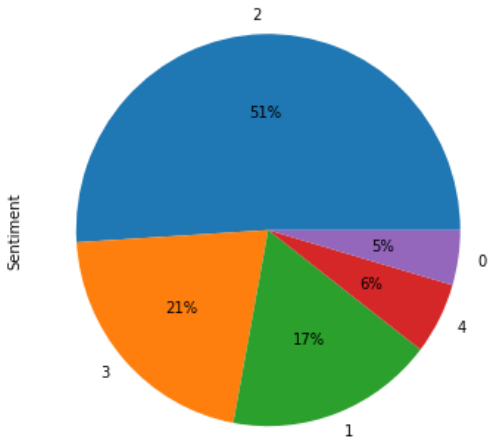
INTRODUCTION:

Sentiment analysis is a widely studied topic by people interested in the field of natural language processing (NLP) and classification models in machine learning. In most cases the data studied for sentiment analysis involves binary classification i.e. a negative and positive sentiment attached to each data point, however in the dataset that I explored in this project has five sentiments. I was inspired to do this project after coming across Stanford's Sentiment Tree visualiser [1].

The problem is taken from the Kaggle [2] competition "sentiment analysis for movie reviews" which also provides us with all the required data that I have worked on in this project in addition to some other data taken from language libraries [3].

DATA:

The data given to us in the Kaggle competition consisted of over 100,000 training sentences each broken down further into phrases. Every phrase is assigned one of the five sentiments (0-negative, 1-somewhat negative, 2-neutral, 3-somewhat positive, 4-positive). The data given has been parsed through the Stanford Parser and has been previously processed to remove repeated phrases from the sentences.



As we can see that the data mostly consists of neutral sentiment phrases (label-2) so we should be expecting to see much better results for the phrases belonging to the neutral category as compared to say phrases in the negative category (label-0) which contribute to just 5% of that data.

Initial distribution of dataset

HURDLES:

The first challenge in dealing with sentiment data is the fact that it is present in text format and must be converted to numerical type in order to get results from the models that we will be using. While doing this step we must also clean the textual data (punctuations, whitespaces, Stop words).

In addition, even though the data given to us is well labelled and free of null/garbage values there is biasing in the predictions due to the sample size of few categories dominating the data. There a few approaches to deal with such sampling which we will have a look at next.

DATA PROCESSING:

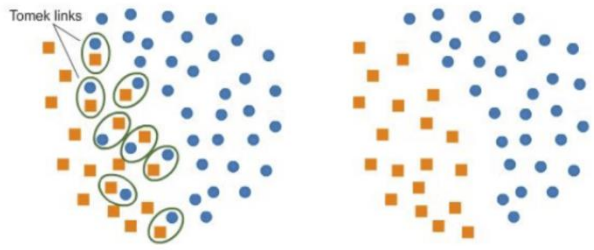
TF-IDF (Term frequency – Inverse Document Frequency): It is one of the most widely used method in text processing. In this method the frequency for each word in the document is calculated (term frequency) then this frequency is offset by the frequency of this word/token in the entire corpus (df). It is a useful method as it balances out the weights of the most commonly and uncommonly occurring words.

$$w_{i,j} = tf_{i,j} \times \log\left(\frac{N}{df_i}\right)$$

$tf_{i,j}$ = number of occurrences of i in j
df_i = number of documents containing i
N = total number of documents

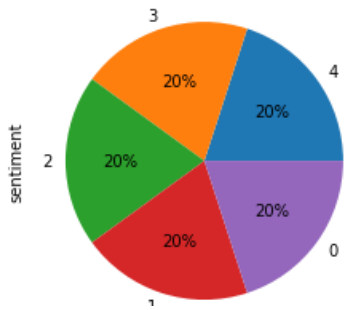
[Figure 1]

Random Under Sampling (RUS): One of the most basic techniques to deal with imbalanced class distribution data samples. It simply removes the values from the majority class or duplicates data in the minority class to match the sample size of the majority class. This method has some significant drawbacks including loss of information.

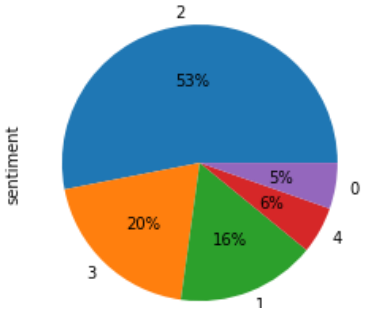


[Figure 2] Tomek links illustration

Tomek Links: Tomek links are used to remove the overlapping data points from the majority class leading to well defined clusters which improves the output of classification models. In our case this method was not very useful due to the small number of overlaps between the majority class and the other classes.



Distribution after Random Under Sampling

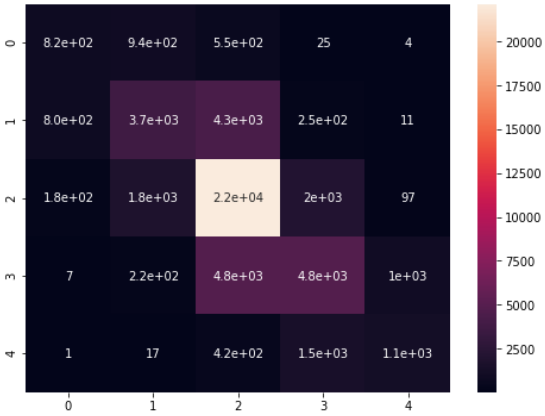


Distribution after Tomek Links

Models Tested and Observations:

Random Forest Classifier:

This classification methodology consists of forming numerous decision trees and bags their features randomly to create a forest of such trees. Ideally the forest would provide better prediction than the trees.



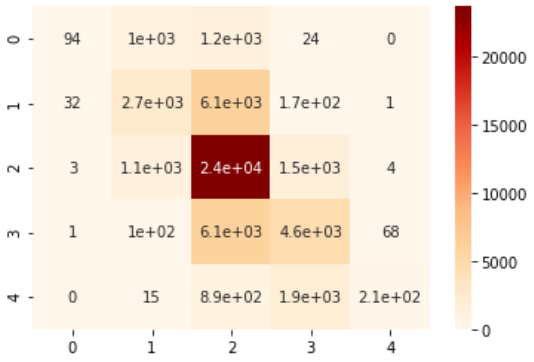
Confusion Matrix for Random Forest Classifier

The random forest classifier has a better distribution in terms of accuracy among classes. The accuracy for 100 estimators came out to be ~66%, by increasing it to 300 estimators the time taken increased significantly with a slight improvement (1-2%) in the results.

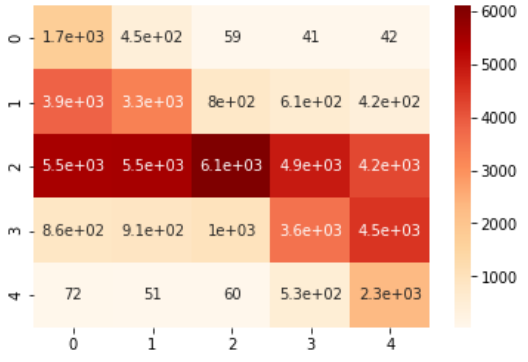
Logistic Regression/ Multinomial Naive Bayes/ Linear SVC:

Both these models are extremely fast and gave surprisingly accurate results. MNB is known for great results when classifying on discrete features (word count in our case). The models had an accuracy of 64% and 61% respectively. They both had an improvement of 1.5-2.0% in their accuracy after applying Tomek links, however the accuracy for the majority sentiment (label-2) dropped after applying RUS but the accuracy for minority classes increased.

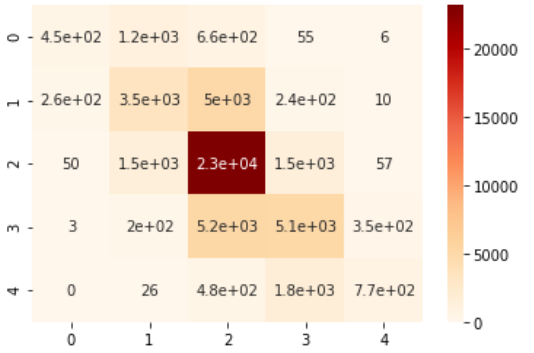
The Linear SVC model gives better accuracy (65%) even among the minority classes without under-sampling.



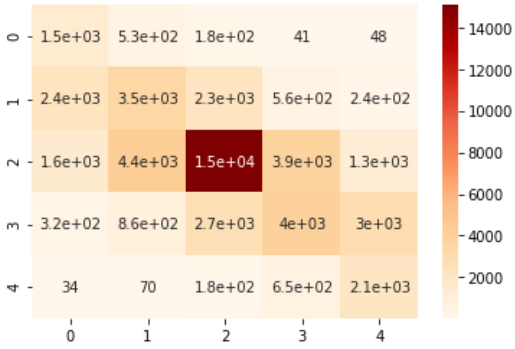
Confusion matrix for MNB before RUS



Confusion matrix for MNB after RUS



Confusion matric for LR before RUS



Confusion matrix for LR after RUS

As we can observe that the accuracy for minority classes (0 and 4) increased by a factor of 1,000 and 100 respectively but the accuracy of the majority class (6) decreased by a factor of 10.

CONCLUSION:

We have observed that the Random Forest Classifier followed by Linear SVC has outperformed other algorithms in the task of classifying sentiments, however the regression model gives decent results considering the time it takes is significantly less than the other models.

This observation is in sync with the previous studies done on the topic of sentiment analysis. Another interesting observation is that Linear SVC models has comparatively better performance across classes even without using RUS.

The output for each model can be found in the results file.

FUTURE WORK:

This study was limited to conventional classification and sampling techniques, the field has evolved a lot and most studies on the topic are done through RNNs and other sophisticated models such as LSTM.

The data can certainly be processed further using techniques like SMOTE which has been been quite effective on solving sampling issues in classification. Additional features and hyper-tuning of models could also result in a better outcome.

REFERENCES:

[1] http://nlp.stanford.edu:8080/sentiment/rntnDemo.html

[2] https://www.kaggle.com/c/sentiment-analysis-on-movie-reviews/overview

[3] https://www.nltk.org/

[Figure 2] https://cdn.searchenginejournal.com/wp-content/uploads/2019/10/screenshot-1.png

[Figure 3] https://www.kaggle.com/rafjaa/resampling-strategies-for-imbalanced-datasets

Libraries and Notebooks that proved helpful in Python implementation:

1. https://imbalanced-learn.readthedocs.io/en/stable/api.html (under-sampling)
2. https://www.kaggle.com/pranjalya/movie-reviews (accuracy calculation of models)
3. https://www.nltk.org/ (language toolkit used for English stop words)
4. https://scikit-learn.org/stable/ (classification models/vectorizer)