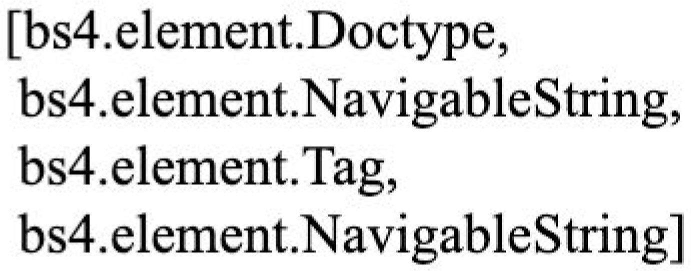
Content:

1. Web scraping
2. Creating json file in specified format
3. Consolidated json files into single dataset
4. Preprocessing text data
5. Bag of Words - CountVectorizer
6. Balancing uneven distribution of classes
7. Keras model
8. Grid search CV
9. Transfer learning
10. Tests on GE
11. Observations

1. Web scraping [1]:
    a. Queried the facebook earnings call website to obtain HTML
    b. Used Beautiful soup python package to parse HTML content
    c. Obtained element information of page



```
[bs4.element.Doctype,
 bs4.element.NavigableString,
 bs4.element.Tag,
 bs4.element.NavigableString]
```

    d. Obtained tags
    e. Obtained paragraphs

2. Creating json file in specified format [2]:
    a. Created a list of numbers for each paragraph
    b. Created json from this dictionary of numbers and parahs
    c. Labelled the sentiments of paragraphs manually

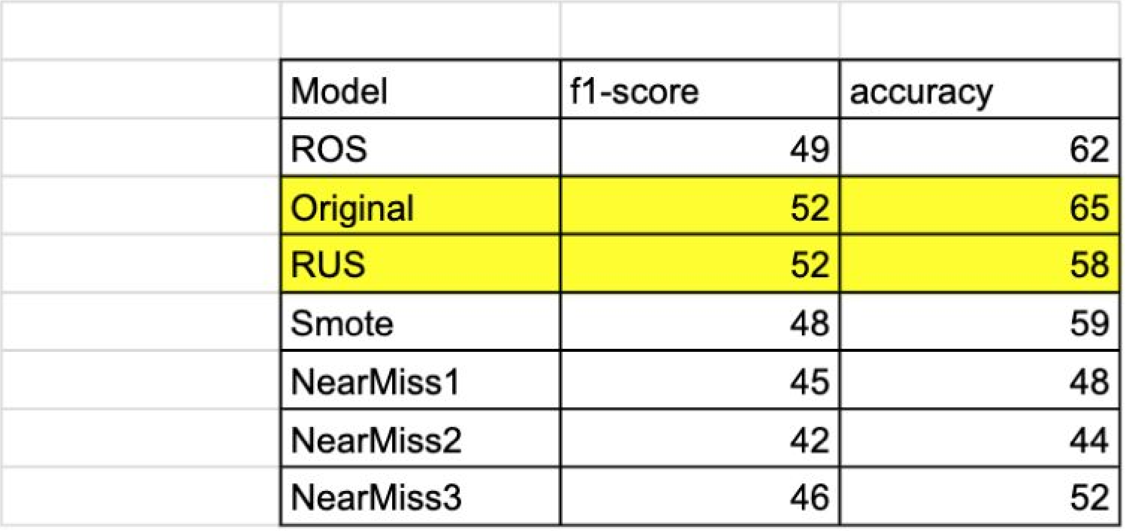
3. Consolidated json files into single dataset [3]:
    a. Collected all 12 json files
    b. Ran a loop to create dictionary
    c. Converted dictionary to dataframe

4. Preprocessing text data [4]:
    a. Implemented NLTK package to remove:
        i. Stop words - the, is, are
        ii. Stemmer to remove inflicted and similar words to roots - like, likes, respons, responsive

        iii. Non-alphabetic characters - !, ?
        iv. Changed to lower case
    b. Tokenized

5. Bag of Words - CountVectorizer [5] :
    a. Sklearn’s CountVectorizer package to create a Bag of Words
    b. Includes maximum 2000 features
    c. Minimum number of occurance of a word to be included in Bag is 3
    d. Maximum frequency is 0.6
    e. Stop words from English language
    f. Total number of words contained = 1649
    g. Created a Logistic Regression model

6. Balancing uneven distribution of classes [a, b, c, d, e, f, g]:
    a. Oversampled: RandomOverSampler - Increased the proportion of negative and positive classes to match that of positive class on the training dataset
    b. Downsampled: RandomUnderSampler - Decreased the proportion of neutral class to match that of positive and negative classes on the training dataset
    c. NearMiss1 - Downsampling with 1 nearest neighbour [ref]
    d. NearMiss2 - Downsampling with 2 nearest neighbour
    e. NearMiss3 - Downsampling with 3 nearest neighbour
    f. SMOTE - Synthetic minority over sampling technique - increase negative and positive class for nearest neighbours to match neutral class
    g. Results -



Model	f1-score	accuracy
ROS	49	62
Original	52	65
RUS	52	58
Smote	48	59
NearMiss1	45	48
NearMiss2	42	44
NearMiss3	46	52

7. Keras model [7]:
    a. Input size of bag of words - 1649, F1 score = 0.48

8. Grid search CV [a , b]:
    a. batch_size = [10, 20, 40, 60, 80, 100] and epochs = [2,5,10, 20], **f1 score = 0.49, Best: 0.692949 using {'batch_size': 80, 'epochs': 5}**
    b. optimizer = ['SGD', 'RMSprop', 'Adagrad', 'Adadelta', 'Adam', 'Adamax', 'Nadam'], **f1 score = 0.52, Best: 0.687642 using {'optimizer': 'Adagrad'}**

9. Transfer learning [i , ii , iii , iv , v]:
    a. Created a bag of words with max features of 2000 on the IMDB dataset (**25000**)
    b. Created bag of words of financial dataset and predicted on the keras model
    c. Created a bag of words with max features of 2000 on the IMDB dataset (**5000**)
    d. Created a bag of words with max features of 2000 on the IMDB dataset (**88585 - all words**)
    e. Max features - 5000

10. Tests on GE [a , b]:
    a. Trained on financial dataset with BoW keras model and tested on GE dataset
    b. Learned on IMDB dataset, tested on whole GE dataset

11. Observations:
    a. Transfer learning gives very bad results - 0.16 f1 score be it on GE or Finance dataset
    b. Adagrad optimizer gives best performance with batch size of 80 and 5 epochs
    c. Balancing does not impact the results a lot, they are very similar to unbalanced results