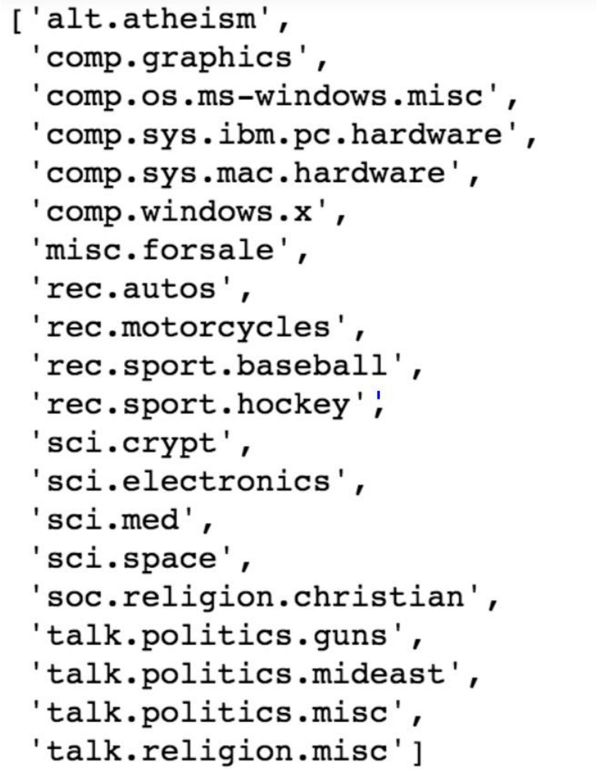
PROJECT 2 REPORT
SHREYAS MOHAN 1001669806

Description

- The whole newsgroup dataset is organized into 20 newsgroups-



```
['alt.atheism',
 'comp.graphics',
 'comp.os.ms-windows.misc',
 'comp.sys.ibm.pc.hardware',
 'comp.sys.mac.hardware',
 'comp.windows.x',
 'misc.forsale',
 'rec.autos',
 'rec.motorcycles',
 'rec.sport.baseball',
 'rec.sport.hockey',
 'sci.crypt',
 'sci.electronics',
 'sci.med',
 'sci.space',
 'soc.religion.christian',
 'talk.politics.guns',
 'talk.politics.mideast',
 'talk.politics.misc',
 'talk.religion.misc']
```

- We use **Naive Bayes classifier** which is suitable for discrete classification.
- Then the model starts with training of first 500 files in each category where the words and its frequency are stored in a dictionary for easy access to test it in the future .
- Since the dataset are series of words, before running any machine learning algorithms, we need to convert text representations into numerical representations. One way to do it is using the "Bag of Words" method, first split each file into single words, then count the number of times each word appears in each file, so every unique word will have a unique value in our dictionary.
- At the end of training model, we will have dictionary of trained files under each category, dictionary of words under each category and a master dictionary of total words.
- Later, the model is tested with the remaining 500 files.

Code Implementation

- It contains a function for data handling named data_handler . First, the input folders are retrieved as a list from the parent directory. This function filters the data by removing the special characters and stop words.

- Stop words are the most frequent words occurring in the file.
- Now, using the trained model we test the remaining 500 files in each category and classify the words by computing log probabilities(because it improves accuracy of model) over the testing data using the formula,

$$P(i \mid j) = \frac{\text{word}_{ij} + \propto}{\text{word}_j + |N| + 1}$$

Where N is the total number of words in vocabulary , word_{ij} counts i in category j, word_j is count of words in category j and ∝ = 0.0001(Laplace smoothing)

$$P(j) = \log \pi_j + \sum_{n=1}^{N} \log(1 + f_i)\log(P(i|j))$$

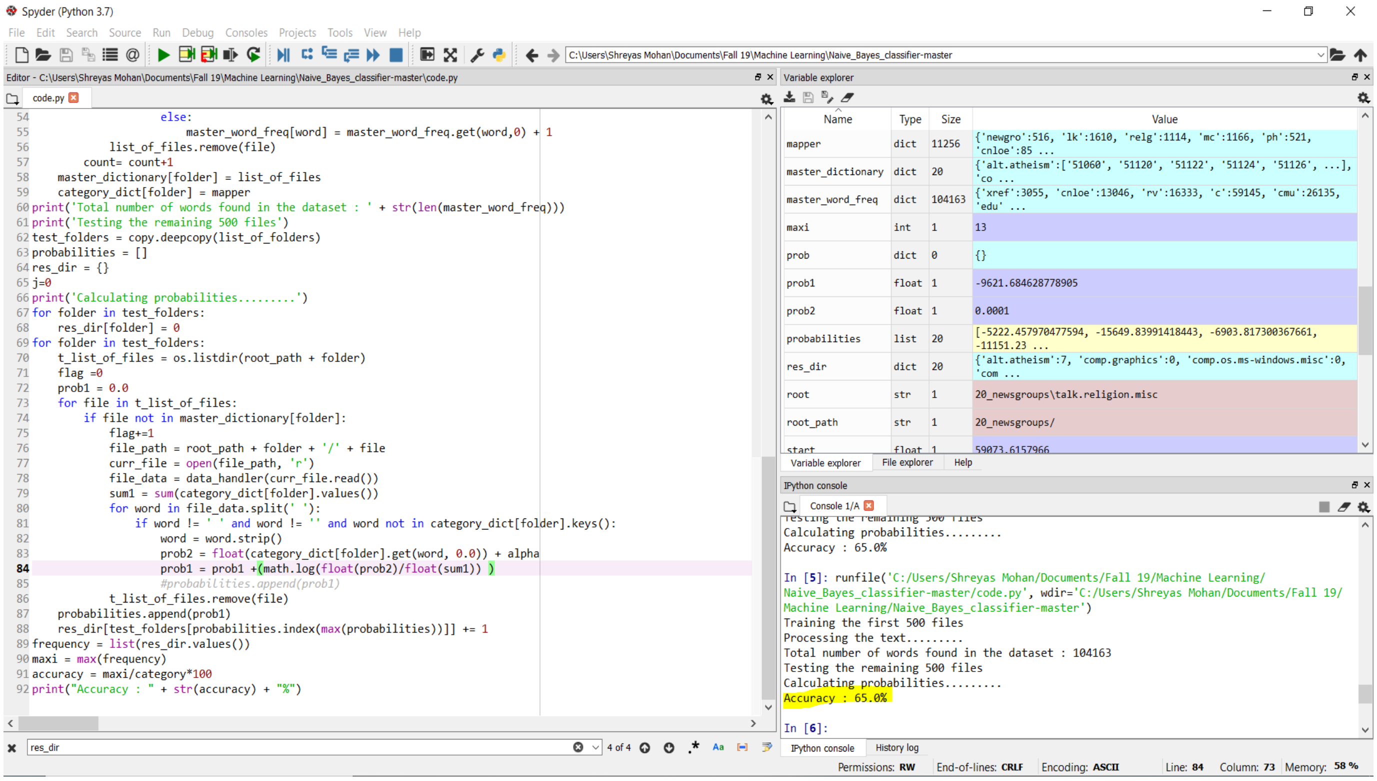
Which is the optimal model.

- The number of collisions of maximum probabilities of a particular word in a category is computed and is divided by the total population in order to calculate accuracy of the model. **Accuracy of this model comes out to be 65.00 % .**

How to improve the performance?

Under-sampling: Remove samples from over-represented classes , use this when have huge dataset like the Newsgroup dataset.

Result Screenshot



```python
                    else:
                        master_word_freq[word] = master_word_freq.get(word,0) + 1
            list_of_files.remove(file)
        count= count+1
    master_dictionary[folder] = list_of_files
    category_dict[folder] = mapper
print('Total number of words found in the dataset : ' + str(len(master_word_freq)))
print('Testing the remaining 500 files')
test_folders = copy.deepcopy(list_of_folders)
probabilities = []
res_dir = {}
j=0
print('Calculating probabilities.........')
for folder in test_folders:
    res_dir[folder] = 0
for folder in test_folders:
    t_list_of_files = os.listdir(root_path + folder)
    flag =0
    prob1 = 0.0
    for file in t_list_of_files:
        if file not in master_dictionary[folder]:
            flag+=1
            file_path = root_path + folder + '/' + file
            curr_file = open(file_path, 'r')
            file_data = data_handler(curr_file.read())
            sum1 = sum(category_dict[folder].values())
            for word in file_data.split(' '):
                if word != ' ' and word != '' and word not in category_dict[folder].keys():
                    word = word.strip()
                    prob2 = float(category_dict[folder].get(word, 0.0)) + alpha
                    prob1 = prob1 +(math.log(float(prob2)/float(sum1)) )
                    #probabilities.append(prob1)
            t_list_of_files.remove(file)
    probabilities.append(prob1)
    res_dir[test_folders[probabilities.index(max(probabilities))]] += 1
frequency = list(res_dir.values())
maxi = max(frequency)
accuracy = maxi/category*100
print("Accuracy : " + str(accuracy) + "%")
```

Name	Type	Size	Value
mapper	dict	11256	{'newgro':516, 'lk':1610, 'relg':1114, 'mc':1166, 'ph':521, 'cnloe':85 ...
master_dictionary	dict	20	{'alt.atheism':['51060', '51120', '51122', '51124', '51126', ...], 'co ...
master_word_freq	dict	104163	{'xref':3055, 'cnloe':13046, 'rv':16333, 'c':59145, 'cmu':26135, 'edu' ...
maxi	int	1	13
prob	dict	0	{}
prob1	float	1	-9621.684628778905
prob2	float	1	0.0001
probabilities	list	20	[-5222.457970477594, -15649.83991418443, -6903.817300367661, -11151.23 ...
res_dir	dict	20	{'alt.atheism':7, 'comp.graphics':0, 'comp.os.ms-windows.misc':0, 'com ...
root	str	1	20_newsgroups\talk.religion.misc
root_path	str	1	20_newsgroups/
start	float	1	59073.6157966

```
Testing the remaining 500 files
Calculating probabilities.........
Accuracy : 65.0%

In [5]: runfile('C:/Users/Shreyas Mohan/Documents/Fall 19/Machine Learning/
Naive_Bayes_classifier-master/code.py', wdir='C:/Users/Shreyas Mohan/Documents/Fall 19/
Machine Learning/Naive_Bayes_classifier-master')
Training the first 500 files
Processing the text.........
Total number of words found in the dataset : 104163
Testing the remaining 500 files
Calculating probabilities.........
Accuracy : 65.0%

In [6]:
```